Desain dan Implementasi *Voice Command* Menggunakan Metode MFCC dan HMMs

Muslim Sidiq[1], Tjokorda Agung Budi W[2], Siti Sa'adah[3]
[1.2.3] Fakultas Informatika, School of Computing, Universitas Telkom
Jalan Telekomunikasi No.1, Dayeuh Kolot, Bandung 40257
muslimsidiq@gmail.com[1], cokagung@telkomuniversity.ac.id [2], tisataz@gmail.com [3]

Abstrak

Semakin seringnya interaksi manusia terhadap teknologi menuntut pengembangan metode interaksi dengan mesin ke arah yang lebih natural. Suara yang merupakan komunikasi yang paling sering digunakan manusia menjadikannya salah satu metode interaksi yang natural. Maka dari itu pengembangan sistem yang dapat mengenali ucapan manusia sebagai suatu aksi pada mesin dapat menjadi satu pilihan untuk permasalahan tersebut. *Voice command* yang merupakan sistem *speech recognition* untuk meberikan fungsi dan aksi pada sistem yang telah didefinisikan sebagai *Command* dan *Control systems*. Nilai amplitudo diambil dari sinyal suara masukan, sehingga didapatkan kumpulan angka *real* yang menjadi nilai masukan untuk ekstraksi ciri. Metode ekstraksi ciri yang digunakan pada tugas akhir ini adalah *Mel Frequency Cepstral Coefficient* (MFCC). Tahapan awal MFCC adalah memecah nilai amplitudo sinyal masukan menjadi *frame-frame* yang diolah dengan menggunakan *mel-filterbak* yang diadaptasi dari cara kerja pendengaran manusia. Hasil ekstraksi ciri kemudian dibuat menjadi *codebook* yang digunakan sebagai inputan simbol pada HMM untuk membentuk model dari setiap kata. Ketika pengujian ciri dari sinyal uji yang telah dikuantisasi kemudian dicocokan dengan model yang telah dibuat pada tahap sebelumnya, sehingga kata dapat dikenali. Dari hasil pengujian, sistem dapat mengenali kata yang diucapkan penutur dengan nilai akurasi rata-rata sebesar 93.89% pada lingkungan tanpa *noise*, dan 58.1% pada lingkungan dengan *noise*.

Kata kunci: *Voice Command*, MFCC, HMMs

Abstract

The more frequent human interaction to technology demands the development of methods of interaction with the machine to a more natural. Sound which is the most frequently used communication humans makes it one of the natural methods of interaction. Thus the development of a system that can recognize human speech as an action on the machine can be an option for those problems. Voice command is a speech recognition system to gave the functions and actions on a system that has been defined as Command and Control systems. Amplitude value is taken from the input sound signal, so we get a set of real numbers is the value input for feature extraction. Feature extraction methods used in this thesis are Mel frequency cepstral coefficient (MFCC). Early stages of MFCC is splitting the input signal amplitude values into frames which are then processed using the mel-filterbak adapted from the workings of the human hearing. The results of feature extraction made into a codebook which is used as an input symbol on HMM to form a model of every word. When testing the characteristics of the test signal that has been quantized and then matched with the model that has been created in the previous step, so that the word can be recognized. From the test results, the system can recognize spoken words of speakers with an average accuracy of 93.89% in the environment without noise, and 58.1% in environments with noise.

Keyword : Voice Command, MFCC, HMMs

1. Pendahuluan

Pengembangan teknologi yang dapat berinteraksi dengan manusia secara natural menjadi salah satu topik yang menjadi tren saat ini. Dengan mengadaptasi beberapa metode komunikasi manusia, maka diharapkan interaksi manusia dengan mesin juga menjadi lebih natural atau bahkan dapat memudahkan pengguna dengan kebutuhan khusus seperti penyandang disabilitas. Beberapa contoh metode interaksi yang ada antara lain sentuhan tangan, gerakan mata, gerakan hidung, pengucapan kata, gerakan kepala dan sebagainya [4].

Penggunaan suara sebagai salah satu metode interaksi pada mesin, memberikan satu solusi untuk permasalahan diatas. Suara yang merupakan komunikasi yang paling sering digunakan manusia menjadikannya salah satu metode yang cukup natural. Metode interaksi tersebut biasa dikenal dengan *voice command* yang merupakan sistem *speech recognition* untuk memberikan fungsi dan aksi pada sistem yang telah didefinisikan sebagai *Command* dan *Control systems* [17]. Pada penerapannya *voice command* dapat digunakan untuk menyalakan lampu, menyalakan televisi, memasukkan nomor telepon, hanya menggunakan suara [10]. Target implementasi *voice command* biasanya pada perangkat yang memiliki fungsi navigasi. Sehingga fungsi kontrol yang seharusnya diberikan pengguna dengan kontak fisik dapat digantikan dengan input suara [14].

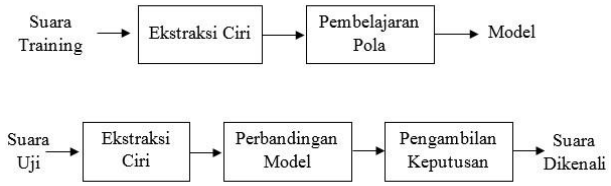
Banyak metode yang telah digunakan pada implementasi *speech recognition* seperti menggunakan MFCC dengan HMM, MFCC dengan VQ, LPC dengan HMM, dan MFCC dengan DTW [3]. Pada tugas akhir untuk mengimplementasi sistem *voice command* digunakan dua metode, yaitu metode ekstraksi ciri dan metode pecocokan model. Untuk metode ekstraksi ciri digunakan *Mel Frequency Cepstral Coefficient* (MFCC) dan *Hidden Markov Models* (HMM) sebagai metode pencocokan. MFCC merupakan salah satu metode ekstraksi ciri yang mudah dan sering digunakan. Kelebihan MFCC dalam mengekstraksi ciri adalah ada pada *mel-frequency* yang mengadaptasi pendengaran manusia dengan cara memfilter secara linier pada frekuensi dibawah 1000Hz dan secara logaritmik pada frekuensi diatas 1000Hz [1]. HHMs sendiri merupakan metode probabilistik yang menyediakan *framework* yang sederhana dan efektif untuk membangun pemodelan sinyal, sehingga cocok untuk pengolahan sinyal suara yang memiliki sifat selalu berubah-ubah berdasarkan waktu [11].

2. Landasan Teori

2.1 *Speech Recognition*

Speech recognition (juga dikenal dengan *Automatic Speech Recognition (ASR)*, atau *Computer Speech Recognition*) dapat diartikan sebagai proses mengubah sinyal suara menjadi urutan kata dengan algoritma yang telah di implementasi sebagai program komputer. *Speech recognition* merupakan cabang dari *speech processing*. Tujuan yang lebih yang lebih khusus adalah membuat komputer dapat mengenali kata yang diucapkan manusia seperti manusia lain mengenalinya. Pada gambar dibawah ini dijelaskan klasifikasi pengolahan kata atau *speech processing* [10].

2.1.1 Tahapan *speech recognition*



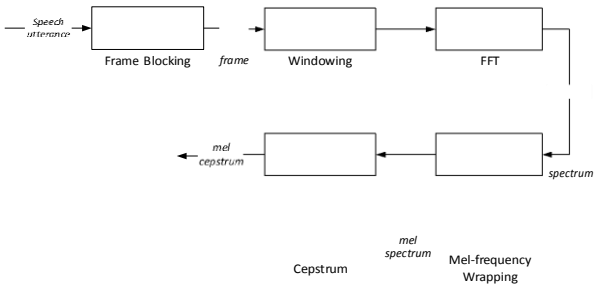
Gambar 2.1 Skema Dasar Speech Recognition [5]

Secara umum *speech recognition* memiliki beberapa tahapan yang pertama adalah pembuatan model. Pembuatan model dilakukan dengan inputan suara *training* kemudian di ekstraksi cirinya. Ciri atau pola yang di dapat kemudian dipelajari sistem sehingga menjadi model. Pembuatan model ini bertujuan untuk mengindentifikasi pola kata yang nanti akan digunakan sebagai pembanding ketika proses pengenalan dilakukan. Dibutuhkan data *training* sebagai *input*, pada kasus ini kita sebut sebagai suara pembelajaran.

2.2 Ekstraksi Ciri Dengan MFCC

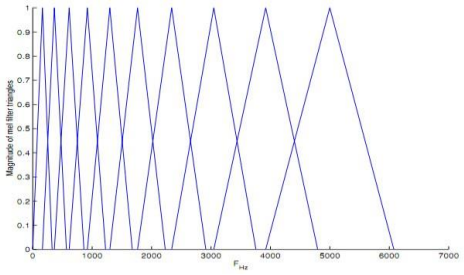
Ekstraksi ciri digunakan untuk mendapatkan ciri dari sinyal suara misalnya frekuensi, *amplitude*, *power*, intensitas dan sebagainya. Ekstraksi dengan parameter ciri terbaik dari sinyal akuistik merupakan tahap penting untuk meningkatkan akurasi pengenalan wicara. Pengolahan (MFCC) *Mel Frequency Cepstral Coefficient* didasarkan pada persepsi pendegaran manusia yang tidak mampu mendengar suara dengan frekuensi lebih dari satu KHz. Maka dari itu MFCC menggunakan dua jenis filter

yaitu mengunakan frekuensi linier di bawah 1000 Hz dan frekuensi logaritmik di atas 1000 Hz. Nilai frekuensi tersebut dikenal dengan *mel-frequency* [5].



Gambar 2.2 Proses MFCC [5]

Pada gambar 2.2 dibawah menujukkan bagaimana MFCC ini membedakan inputan suara pada frekuensi rendah dan tinggi. Untuk menangkap suara dengan frekuensi rendah, metode ini akan meningkatkan sesitifitas. Bisa dilihat dengan semakin pendeknya jarak antar segitiga. Begitu sebaliknya untuk suara dengan frekuensi tinggi.



Gambar 2.3 Contoh Mel Filter Bank. [17]

2.2.1 Tahapan *Mel Frequency Cepstral Coefficient*

Ekstraksi ciri dengan MFCC sendiri memiliki beberapa tahapan yaitu [17] [15]:

a. *Frame Blocking*

Proses *framming* digunakan untuk memotong-motong sinyal suara dengan durasi yang panjang menjadi menjadi sinyal suara dengan durasi yang lebih kecil sehingga didapatkan karakteristik sinyal suara yang lebih stabil. Sinyal suara manusia merupakan sinyal yang tidak stabil, dimana tinggi rendah, intensitas, dan koefisien lain selalu berubah. Tetapi dapat diasumsikan sinyal suara dengan panjang 10-30 *millisecond*(ms) bersifat stabil.

b. *Windowing*

menggenerate sinyal ke angka nol pada awal dan akhir setiap *frame. framming* dilakukan pada sinyal kontinu sedangkan *Windowing* dilakukan pada sinyal diskrit.

$$w(k) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi k}{K-1}\right) \quad (2\text{-}1)$$

N = Jumlah sampel, n = indeks window, K=jumlah *frame*

c. *Fast Fourier Transform* (FFT)

Untuk mendapatkan sinyal dalam domain frekuensi dari sebuah sinyal diskrit, salah satu metode *transformasi fourier* yang digunakan adalah *Fast Fourier Transform* (FFT). FFT dilakukan terhadap masing-masing *frame* dari sinyal yang telah di-windowing. FFT merupakan algoritma *Discrete Fourier Transform* (DFT) versi cepat. Yang dioprasikan pada sinyal diskit yang terdiri dari N sampel.

$$\text{[illegible]} \quad (2\text{-}2)$$

N = Jumlah sampel, n= indeks sampel, y = sinyal hasil *windowing*

d. *Mel-Frequency Wrapping*

Skala *Mel-Frequency* adalah frekuensi yang linier di bawah 1 kHz dan logaritmik di atas 1 kHz. Di mana B adalah skala *Mel-Frequency* dan f adalah frekuensi linier. Skala *Mel* dapat diperoleh dengan persamaan :

$$\text{[illegible]} \quad (2\text{-}3)$$

Sedangkan pada *windowing* akan memproses hasil dari *framing* yang berupa potongan-potongan kecil dari sinyal suara dengan meminimalisir diskontinuitas pada bagian awal dan akhir potongan. Konsepnya adalah

f = frekuensi *linear*(Hz), B (f) = skala *Mel-frequency*

Dalam mel-frequency wrapping, sinyal hasil FFT dikelompokkan ke dalam berkas *filter triangular* ini. Maksud pengelompokan di sini adalah setiap nilai FFT dikalikan terhadap *gain filter* yang bersesuaian dan hasilnya dijumlahkan.

e. *Mel-Frequency Cepstrum*

Mel-Frequency Cepstrum merupakan kebalikan dari *spectrum*. Pada langkah ini, spektrum log *mel* dikonversi menjadi *cepstrum* menggunakan *Discrete Cosine Transform* (DCT) untuk mendapatkan kembali sinyal dalam domain waktu. Hasilnya disebut sebagai *Mel-Frequency Cepstral Coefficient* (MFCC). MFCC bisa didapat dari pendekatan persamaan :

$$cep_s(n;m) = \sum_{k=0}^{N-1} \alpha_k \cdot \log(fmel_k) \cos\left(\frac{\pi_{(2n+1)k}}{2N}\right) \qquad (2\text{-}4)$$

N = jumlah sampel , n = 0,1,2,3…N-1, *fmel* = frekuensi *mel*

Di mana cep_s adalah hasil akumulasi dari kuadratik magnitude DFT yang dikalikan dengan *Mel-Filter Bank*. Setelah itu didapatlah MFCC. Pada sistem pengenalan suara, biasanya hanya 13 cepstrum koefisien pertama yang digunakan.

2.2.2 Pengukuran *Energy*

Merupakan salah satu cara untuk menambah nilai koefisien yang dihitung dari *linear prediction* atau *mel-cepstrum*, nilai tersebut merupakan *log energy signal*. Ini berarti pada setiap *frame* terdapat nilai energi yang ditambahkan, berikut rumus untuk menghitung nilai energi :

$$\text{energi} = \log \sum_{n=0}^{m-1} x\_windowed(n)^2 \qquad (2\text{-}5)$$

x_windowed = sinyal hasil *windowing*, k = jumlah *frame*, m = panjang *frame*

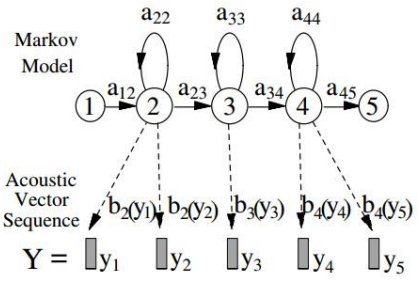
2.2.3 *Delta* dan *Delta-delta feature*

Secara umum metode yang digunakan untuk mandapatkan informasi dari ciri yang dinamis biasa disebut dengan *delta-features*. Turunan waktu dari ciri dapat dihitung dengan beberapa metode, hasil dari perhitungan *delta* akan ditambahkan ke vektor ciri, sehingga menghasilkan vektor ciri yang lebih besar. Nilai dari *delta* akan diturunkan sekali lagi terhadap waktu menjadi nilai *delta-delta* pada beberapa kasus *delta-delta* disebut dengan koefisien percepatan, karena nilai tersebut turunan dari kuadrat waktu dari koefisien.

2.3 *Hidden Markov Models*

Hidden Markov Model (HMM) merupakan pendekatan dengan menggunakan kumpulan *state* yang dimana *state – state* ini memiliki hubungan satu sama lain yang menggambarkan suatu kejadian di dunia nyata, dengan *layer* observasi sebagai hal yang dapat diamati, dan *layer* tersembunyi sebagai *layer* yang hanya bisa diamati melalui *layer* observasi. HMM ini dapat memodelkan persoalan – persoalan di dunia nyata yang sifatnya probabilistik.

Terdapat bebera tipe HMM berdasarkan struktur *state*-nya, antara lain tipe *ergodic*, *left-to-right* dan *modification left-to-right*. Pada tugas akhir ini digunakan tipe *left-to-right*, tipe tersebut digambarkan pada gambar dibawah ini.



Gambar 2.4 HMM- berbasis model suara. [11]

2.3.1 Tiga Permasalahan pada HMM

Dalam implementasi pada *speech recognition*, HMM memiliki tiga permasalahan yang mendasar, yaitu [2] [15]:

a. Masalah petama - *Recognition*

Pada permasalahan HMM yang pertama ini berkaitan dengan evaluasi model atau bisa dikatan pemberian nilai untuk tiap parameter dalam model HMM. Dengan adanya rangkaian observasi $O = (o_1, o_2, \ldots o_T)$ dan model $\lambda = (A, B, \pi)$, bagaimana menentukan nilai probabilitas dari model yang ditentukan (P (O | λ)), dan bagaimana menghitung efisiensinya.

b. Masalah kedua - *Optimal state sequence*

Pada permasalahan HMM yang kedua ini berkaitan dengan *decoding* atau bisa disebut sebagai pemilihan *state* – sate yang optimal. Jika diberikan rangkaian observasi $O = (o_1, o_2, \ldots o_T)$ dan model λ, bagaimana menentukan *state sequence* yang optimal dalam mereprestasikan rangkaian observasi.

c. Masalah ketiga - *Adjustment*

Pada permasalahan HMM yang ketiga ini berkaitan dengan training atau bisa disebut dengan pengaturan parameter dari model. Bagaimana menentukan probabilitas model λ=(A,B,π) agar mendapatkan nilai probabilitas P(O | λ) yang maksimal.

2.3.2 Solusi Permasalahan HMM

Berikut adalah pemaparan solusi untuk memecahkan ketiga permasalahan dasar HMM yang telah dijelaskan sebelumnya [2]:

a. Solusi Masalah Pertama

Permasalahan ini diatasi dengan menggunakan metode *forward-backward algorithm*. Sebenarnya metode ini adalah penggabungan dari dua metode, yaitu metode *forward algorithm* dan metode *backward algorithm*.

Jika kita ingin menghitung probabilitas urutan observasi, $O = \{o_1, o_2, \dots, o_t\}$ terhadap model λ, maka cara yang paling mudah untuk menyelesaikan permasalahan ini adalah melalui enumerasi semua kemungkinan barisan *state* dengan panjang t (banyaknya *state* observasi).

Prosedur *Forward*

Misalkan terdapat suatu variabel probabilitas *forward* pada waktu ke-*t* dan *state* ke-*i* yang dinotasikan dengan $\alpha_t(i)$ dimana secara matematis :

$$\alpha(i) = P(o_1 o_2 \dots o_t, q_t = i|\lambda) \quad (2\text{-}6)$$

$\alpha(i)$ = probabilitas *forward*, P = probabilitas observasi, o = observasi

Prosedur *Backward*

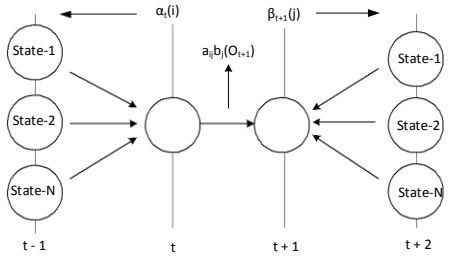
Metode ini disebut prosedur *backward* karena perhitungan nilai probabilitas $P(O|\lambda)$ dicari secara mundur (*backward)* dari waktu observasi maksimum (t = T) sampai dengan waktu observasi awal (t = 1). Misalkan terdapat suatu variabel probabilitas *backward* pada waktu ke-*t* dan pada *state* ke-I yang dinotasikan dengan $\beta_t(i)$ dimana secara matematis :

$$\beta(i) = P(o_{t+1} o_{t+2} \dots o_T, q_t = i|\lambda) \quad (2\text{-}7)$$

$\beta(i)$ = probabilitas *backward*, P = probability observasi, o = observasi.

Prosedur *Forward – Backward*

Nilai $P(O|\lambda)$ dapat dihitung menggunakan prosedur gabungan antara fungsi probabilitas *forward* $\alpha(i)$ dan fungsi probabilitas *backward* $\beta(i)$. Ilustrasi prosedur ini dapat dilihat pada gambar di bawah ini :



Gambar 2.5 Prosedur *forward backward*

Peluang berada di *state* ke-I pada waktu t dari N buah *state* sebelumnya pada waktu t-1 dihitung menggunakan fungsi probabilitas forward $\alpha_t(i)$. Perluang transisi dari *state* ke-I pada waktu t menuju *state* ke-j pada waktu t+1 dan mengambil sebuah simbol observasi pada *state* ke-j tersebut adalah $a_{ij}b_j(o_{t+1})$. Fungsi probabilitas *backward* $\beta_t(i)$ digunakan untuk menghitung peluang munculnya deretan simbol observasi yang dimulai dari waktu *t +1* sampai T.

Secara matematis, perhitungan $P(O|\lambda)$ menggunakan prosedur *forward-backward* dinyatakan sebagai berikut :

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha(i)\beta(i) \quad (2\text{-}8)$$

$\beta(i)$ = probabilitas *backward* ke-i, $\alpha(i)$ = probabilitas *forward* ke-i,

b. Solusi Masalah Kedua

Tidak seperti pada permasalahan pertama yang menghasilkan sebuah solusi, permasalahan kedua atau permasalahan *decoding* HMM bisa diselesaikan dalam berbagai cara. Karena tidak ada solusi yang paling benar dalam pencarian solusi dari masalah *decoding*, sehingga kita akan mencari barisan *state* solusi yang paling optimal berdasarkan barisan observasi yang diberikan. [2]

Algoritma *viterbi* merupakan metode untuk mencari deretan *state* yang terbaik, $Q = \{q_1, q_2, \dots q_t\}$ dengan barisan observasi $O = \{o_1, o_2, \dots, o_t\}$, kita perlu mencari kuantitas dari [15] :

$$\delta(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1 q_2 \dots q_t = i, o_1 o_2 \dots o_t|\lambda] \quad (2\text{-}9)$$

$\delta_t(i)$ = probabilitas yang terbaik, q = *state*

Disebutkan bahwa $\delta_t(i)$ merupakan nilai yang terbaik (probabilitas tertinggi) yang didapatkan, pada waktu t,

yang menghitung mulai dari t pertama observasi dan berakhir di *state* S_i. Dengan menggunakan induksi matematika, kita mendapatkan :

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] \cdot b_j(O_{t+1}) \qquad (2\text{-}10)$$

$\alpha_{t+1}(j)$ merupakan nilai untuk *state* selanjutnya atau *next state*.

c. Solusi Masalah Ketiga

Masalah ketiga ini adalah menyesuaikan parameter-parameter (A, B, π) berdasarkan kriteria optimal tertentu. Metode yang biasa digunakan untuk memecahkan masalah ketiga ini adalah algoritma *Baum-Welch* atau sering disebut sebagai algoritma *forward – backward*. Algoritma ini merupakan suatu metode *iterative* yang berfungsi untuk mencari nilai – nilai maksimum lokal dari fungsi probabilitas $P(O|\lambda)$. Proses *training* ini berlangsung terus sampai kondisi minimum local terpenuhi. Model hasil *training* harus lebih baik daripada model sebelumya.

Formula *Baum-Welch re*-estimasi untuk mean dan kovarian pada masing masing *state* HMM adalah [15]:

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \gamma_t(j) O_t}{\sum_{t=1}^{T} \gamma_t(j)} \qquad (2\text{-}11)$$

$$\hat{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(O_t - \mu_j)(O_t - \mu_j)'}{\sum_{t=1}^{T} \gamma_t(j)} \qquad (2\text{-}12)$$

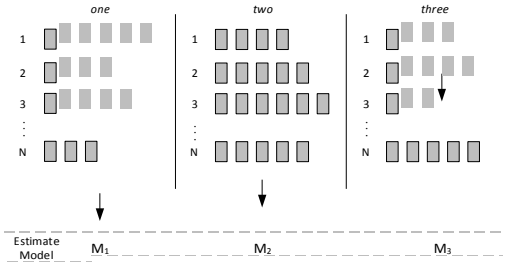
L_j = probabiliti *state* j, μ = *mean*, Σ = nilai kovarian

Parameter vektor akan diestimasi dengan menggunakan algoritma *forward-backward* hingga diperoleh nilai probabilitas P(O|M) terbesar berdasarkan observasi pada masing masing *state*.

Algoritma untuk membentuk re-estimasi parameter HMM dengan *BaumWelch re*-estimasi adalah sebagai berikut :

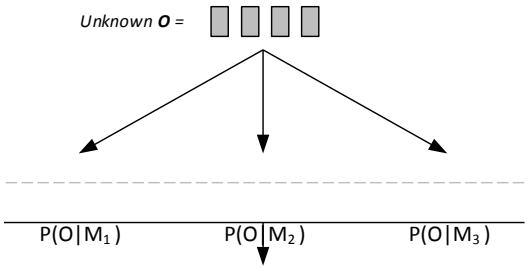
1. Untuk setiap vektor parameter/matrik, alokasikan storage untuk pembilang dan penyebut formula *Baum-Welch* sebagai akumulator.
4. Gunakan nilai akumulator terakhir untuk menghitung nilai parameter yang baru.
5. Jika nilai P = P(***O***|M) iterasi saat ini kurang dari iterasi sebelumnya maka berhenti jika tidak ulangi langkah diatas dengan menggunakan nilai parameter yang baru.

Berikut contoh proses pembelajaran untuk rangkaian observasi :



Gambar 2.6 Proses pembelajaran model [2]

Berikut contoh proses pengenalan untuk rangkaian observasi :



Gambar 2.7 Proses pengenalan

2. Hitung kemungkinan *forward* dan *backward* untuk semua *state j* pada waktu *t*.
3. Untuk setiap *state j* dan waktu *t*, gunakan probabiltas $L_t(t)$ dan vektor observasi saat ini untuk merubah akumulator pada *state* itu.

Pada tahap awalnya HMM melakukan pembelajaran untuk memodelkan beberapa contoh kata, dalam hal ini adalah *"one, two, three*". Hasil dari pembelajaran adalah model yang telah dilakukan estimasi(M). Kemudian HMM digunakan untuk mengenali kata/observasi (O) berdasarkan hasil pembelajaran tersebut. P(O|M) adalah kemungkinan rangkaian observasi O terhadap model M.

4. Pengujian Sistem dan Analisis

4.1 Pengujian Sistem

Pengujian diperlukan untuk mengetahui apakah sistem sudah mampu berjalan sesuai dengan tujuan atau belum. Dan analisi digunakan untuk mengetahui parameter-parameter apa saja yang berpengaruh pada kerja sistem dengan memberikan beberapa variasi nilai parameter sehingga performansi sistem dapat meningkat. Bab ini akan menjelaskan bagaimana skenario, hasil dan analisis

4.2 Persiapan *Dataset*

4.2.1 Pemilihan Kamus Kata

Kata yang dipilih dalah tugas akhir ini merupakan beberapa kata perintah dan kata digit angka yang diucapkan dalah bahasa inggris. Setiap kata memiliki fungsi tersendiri dalam memberikan perintah atau bahkan fungsi tersebut dapat didefinisikan sendiri oleh user. Berikut 10 kata yang akan digunakan dalam tugas akhir ini :

Tabel 4.1 Daftar dictionary kata

No	Kata	No	Kata
1	*Up*	**6**	*One*
2	*Down*	**7**	*Two*
3	*Stop*	**8**	*Three*
4	*Start*	**9**	*Four*
5	*Back*	**10**	*Five*

4.2.2 Pembuatan Dataset Sintetis

Dataset sintetis yang digunakan pada penelitian ini berupa *file wav* dengan *sampling rate* 44100, 16 bit, dengan *channel mono*. Dataset terdiri dari 10 kata yang merupakan hasil *generate* dari aplikasi *text to speech* diunduh dari *website* www.acapela-group.com. *File wav* yang didapatkan merupakan *file* yang bersih dari *noise* suara, sehingga akan menjadikan proses ekstraksi lebih baik.

Terdapat lima penutur dengan jenis kelamin yang berbeda, dari kata tersebut akan divariasikan nilai *pitch*, *volume*, dan *speed*, sehingga akan di dapatkan 4 *file* baru.
Maka terdapat 25 *file* suara dalam satu kata. Berikut aturan

Tabel 4.2 Variasi parameter *file wav*

FILE	PITCH	VOLUME	SPEED
kata_01.wav	0	0	0
kata_02.wav	-1	+10	+10
kata_03.wav	-2	+20	+20
kata_04.wav	+1	-10	-10
kata_05.wav	+2	-20	-20

4.2.3 Perekaman Dataset

Perekaman *dataset* ini merupakan perekaman dari penutur asli, yang dilakukan diruangan dengan *noise* sedang sampai dengan tinggi. Proses ini bertujuan untuk mengambil dataset pada kondisi *real*. Perekaman oleh 30 orang penutur laki-laki dan perempuan. Setiap kata diucapkan sebanyak 5 kali, sehingga akan didapatkan 150 data *file* suara pada setiap kata.

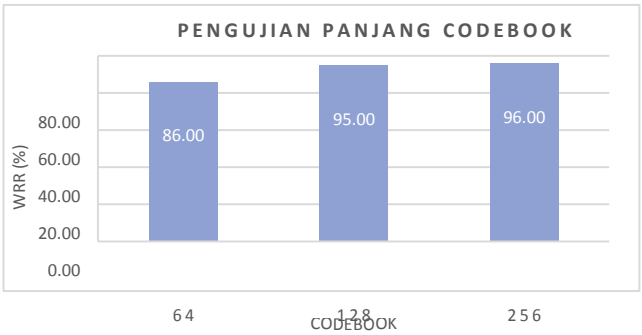
4.3 Sekenario Pengujian

Dalam penelitian ini dilakukan pengujian berdasarkan parameter dengan tujuan untuk mengetahui pengaruh parameter tersebut terhadap tingkat akurasi sistem. ***Codebook*=256**, ***State* HMM = 6** dan ***Sampling rate* =44100** ditetapkan sebagai nilai *default*, sehingga ketika sebuah parameter diuji, parameter yang lainnya akan diisi dengan nilai *default* tersebut. Rumus untuk menghitung tingkat akurasi sistem dijelaskan pada bagian **2.5**. yaitu dengan menggunakan rumus *word recognition rate* (WRR) yang akan membandingkan jumlah kata yang dikenali dengan jumlah kata yang diuji.

4.4 Analisis Hasil Pengujian

4.4.1 Pengujian Panjang *Codebook*

Dari grafik 4.1 dibawah dapat dilihat bahwa peningkatan panjang *codebook* mengakibatkan kecenderungan peningkatan nilai akurasi sistem dalam mengenali kata. Dapat dilihat terjadi peningkatan yang cukup besar dari panjang *codebook* 64 ke 128. Satu titik diwakilkan oleh satu nilai panjang pada *codebook*. Jadi jika *codebook* dengan panjang 256 maka akan terdapat 256 titik, dengan setiap titik memiliki 39 dimensi. Sehingga semakin banyak titik yang digunakan maka akan menambah nilai ciri data yang diolah.

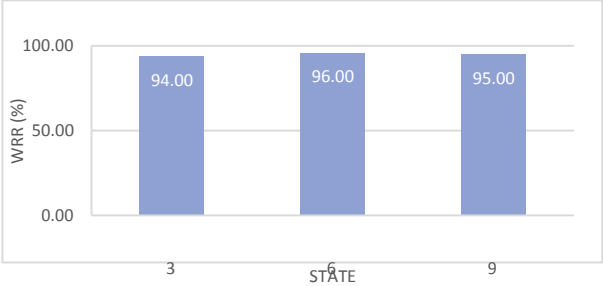


Gambar 4.1 Grafik perbandingan panjang codebook terhadap nilai akurasi

4.4.2 Pengujian Jumlah State HMM

Dari hasil dapat disimpulkan bahwa penambahan jumlah *state* tidak menjamin bahwa nilai akurasi akan meningkat, karena semakin banyak jumlah *state*, maka semakin banyak pula kemungkinan perpindahan dari suatu *state* ke

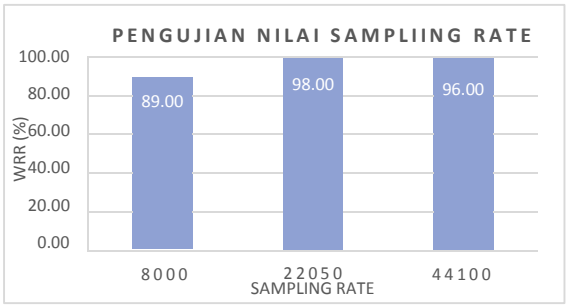
state yang lain, sehingga jumlah *state* yang terlalu banyak dapat menurunkan nilai akurasi karena memperbesar kesalahan dalam pembuatan model, dan jika jumlah *state* yang digunakan terlalu sedikit juga akan menurunkan akurasi sistem.



Gambar 4.2 Grafik perbandingan jumlah state HMM terhadap nilai akurasi

4.4.3 Nilai Sampling Rate

Pada grafik hasil pengujian dengan mengganti nilai *sampling rate* file audio yang digunakan, terlihat bahwa terdapat pengaruh nilai akurasi sistem dengan nilai *samping rate* yang digunakan. Nilai akurasi bertambah sekitar 9% ketika nilai *sampling rate* berubah dari 8000 ke 22050. Dan terjadi sedikit penurunan 2% ketika digunakan *sampling rate* 44100. Semakin besar *sampling rate* nilai pengambilan ciri yang diolah juga semakin banyak sehingga, karena semakin banyaknya nilai ciri yang diolah maka nilai akurasi menjadi turun, namun jika *sampling rate* terlalu kecil akan ada ciri yang hilang.



Gambar 4.3 Grafik perbandingan nilai sampling rate terhadap nilai akurasi

4.4.4 Pengujian Data Asli

Pada sistem audio, terdapat banyak sumber *noise* yang dapat mengganggu *output* ideal dari sistem audio tersebut. *Noise* yang mungkin terjadi pada sistem audio adalah *noise* akustik, *Noise* merupakan suara yang berasal dari sumber lain di sekitar sistem tersebut, seperti suara kipas angin atau suara deru kendaraan yang melintas.

Hasil pengujian data asli didapatkan akurasi sistem sebesar **58.2%.** penurunan akurasi ini disebabkan karena adanya *noise* yang ikut pada proses perekaman ucapan.

ekstraksi ciri juga menjadi tidak sempurna. Ciri yang tidak sempurna akan berpengaruh pada tahap pencocokan, sehingga hasil pengenalan juga menjadi berkurang. Hal ini membuktikan bahwa tidak hanya kualitas *file* audio yang bagus tapi juga lingkungan ketika dilakukan perekaman juga harus dalam kondisi yang sesuai.

Semakin besar *noise* yang ada dalam sinyal suara maka semakin besar juga pengaruh dalam penurunan nilai akurasi.

4.4.5 Pengujian Pada Android

Pada pengujian ini dilakukan pada perangkat mobile android, dimana sinyal suara hasil rekaman langsung dilakukan pengenalan. Pengujian ini bertujuan untuk mengetahui bagaimana sistem dapat berjalan pada perangkat *mobile* android dan mengetahui tingkat akurasinya. Lingkungan pengujian memiliki *noise* rendah sampai dengan sedang. Terdapat dua jenis pengujian, yang pertama penutur diberikan kesempatan tiga kali untuk menguji setiap kata, jika dari ketiga kesempatan tersebut terdapat minimal satu kali benar, maka pengujian dikatakan benar. Kedua penutur hanya diberikan satu kali kesempatan, sehingga jika dalam satu kali pengujian menghasilkan nilai salah maka pengujian tersebut dikatakan salah. Pengujian dilakukan oleh 30 penutur dengan variasi umur dan gender yang berbeda-beda, untuk lebih jelasnya dapat dilihat pada tabel 4.3.

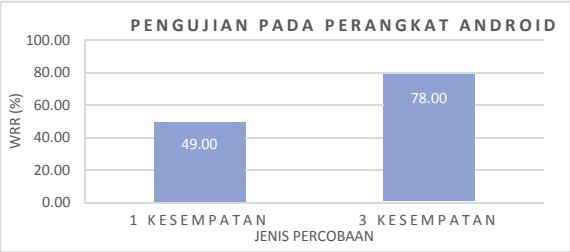
Berikut perhitungan akurasi dengan tiga kali percobaan :

$$\text{Result} = \begin{cases} \text{benar} & 3 \geq T \geq 1 \\ \text{salah} & T = 0 \end{cases}$$

Result = hasil pengujian, T = kemunculan nilai benar

Pada pengujian terakhir menunjukkan menghasilkan nilai akurasi sebesar **49%** dengan kesempatan pengucapan hanya satu kali, dan didapatkan hasil **78%** dengan kesempatan pengucapan tiga kali, hal ini disebabkan karena banyak faktor seperti jarak sumber suara dengan

microfon yang berbeda-beda, noise yang tanpa sengaja ikut dalam proses perekaman.



Gambar 4.4 Grafik pengujian pada perangkat android

5. Penutup

5.1 Kesimpulan

Berdasarkan hasil analisis terhadap pengujian yang dilakukan pada sistem, maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Sistem *voice command* berhasil diimplementasi dengan menggunakan algoritma MFCC sebagai ekstraksi ciri dengan digabungkan dengan *Hidden Markov Model* (HMM) sebagai algoritma pencocokan.
2. Sistem menghasilkan nilai akurasi rata-rata sebesar 93.89% pada lingkungan tanpa *noise*, dan 58.1% pada lingkungan dengan *noise*.
3. Panjang codebook, jumlah *state* HMM, nilai *sampling rate* dan *noise* memberikan pengaruh terhadap akurasi sistem.
4. Pada pengujian sistem, codebook dengan panjang 256, jumlah *state* HMM sama dengan 6, dan nilai *sampling rate* 22050 Hz menghasilkan nilai akurasi paling besar, yaitu 98%.
5. Dari pengujian terakhir (bagian 4.5.4) dapat disimpulkan bahwa sistem berjalan cukup baik pada perangkat android.

5.2 Saran

Pengembangan lebih lanjut yang dapat dilakukan terhadap tugas akhir ini adalah sebagai berikut :

1. Level pengenalan bisa dikembangkan menjadi level *phoneme* sehingga kata yang bisa lebih banyak.
2. Pengembangan yang dapat dilakukan adalah dengan mengimplementasi sistem pada *assitive tool* atau bisa disebut perangkat untuk membantu penyandang disabilitas.
3. *Modals* untuk memberikan perintah (*command*) dapat digabungkan dengan modal yang lain, seperti *gesture* kepala atau mata. Sehingga beberapa aksi yang tidak bisa berikan dengan suara dapat diberikan oleh *modals* yang lain.

Daftar Pustaka

[1] A. K. B. ANJALI BALA, “Voice Command Recognition System Based On Mfcc And Dtw,” dalam *International Journal of Engineering Science and Technology*, 2010.

[2] A. Ahkam, Analisis Penggunaan Hybrid Statistik dan Grammar Hidden Markov Model pada Peringkasan Artikel, Bandung: Institute Teknologi Telkom, 2012.

[3] T. S. G. Ahmad A. M. Abushariah, “English Digits Speech Recognition System Based on Hidden Markov Models,” dalam *International Conference on Computer and Communication Engineering(ICCCE 2010)*, Kuala Lumpur, Malaysia, 2010.

[4] A. K. Andrey Ronzhin, “Assistive Multimodal System Based On Speech Recognition And Head Tracking,” dalam *SPIIRAS*, St. Petersburg, Russia.

[5] A. H. R. R. I. Angga Setiawan, “Aplikasi Pengenalan Ucapan dengan Ekstraksi Mel-Frequency Cepstrum Coefficients (MFCC) Melalui Jaringan Syaraf Tiruan (JST) Learning Vector Quantization (LVQ) untuk Mengoperasikan Kursor Komputer,” *TRANSMISI*, pp. 82-86, 2011.

[6] A. K. S. Ashish Kumar Panda, Study Of Speaker Recognition Systems, ROURKELA: NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA, 2011.

[7] K. R. S. A. Devesh Nema, Implementation Of Speech Recognition In Resource Constrained Environments, INDIAN INSTITUTE OF TECHNOLOGY ROORKEE, 2006.

[8] T. Kinnunen, "Spectral Features for Automatic Text-Independent Speaker Recognition," *University of Joensuu*, 2003.

[9] F. Lawrence R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in

Speech Recognition," dalam *Proceeding of the IEEE, vol. 77, no. 2.*, 1998.

[10] S. M.A.Anusuya, "Speech Recognition by Machine: A Review," in *(IJCSIS) International Journal of Computer Science and Information Security*, 2009.

[11] S. Y. Mark Gales, "The Application of Hidden Markov Models in Speech Recognition," dalam *Foundations and Trends in Signal Processing*, 2007.

[12] K. H. Marunta, Phoneme-Based Speech To Text Berbahasa Indonesia Menggunakan Transformasi Fourier Dan Hidden Markov Model, Banudung: Telkom University Library, 2010.

[13] S. Mashuri Hasan, "Mplementasi Dan Analisis Pengenalan Ucapan Berbasis Fonem Untuk Lingkungan Berderau Pada Perangkat Bergerak Berbasis Android," *Fakultas Informatika Institut Teknologi Telkom, Bandung*, pp. 1-6, 2011.

[14] M. E. Mikael Nilsson, Speech Recognition using Hidden Markov Model(performance evaluation in noisy environment), Sweden: Blekinge Institute of Technology, 2002.

[15] B. S. Nick Bardici, Speech Recognition using Hidden Markov Model, Blekinge Institute of Technology, 2006.

[16] B. W. P. Y. Santosh K.Gaikwad, "A Review on Speech Recognition Technique," dalam *International Journal of Computer Applications (0975 – 8887)*, 2010.

[17] M. Zahit, Robot Control With Voice Command, Istanbul: Yıldız Technical University, Department of Computer Engineering , 2008 .