BAB II

LANDASAN TEORI

2.1 Voice Recognition

Voice recognition terdiri dari dua yaitu *speech recognition* dan *speaker recognition* (Hermsen, 2000).

2.1.1. Speech Recognition

Speech recognition atau pengenalan suara adalah suatu pengembangan teknik dan sistem yang memungkinkan komputer untuk menerima masukan berupa kata yang diucapkan (Dimasatria dkk., 2016). Hasil dari identifikasi kata yang diucapkan dapat ditampilkan dalam bentuk tulisan (Dimasatria dkk., 2016).

2.1.2 Speaker Recognition

Speaker recognition adalah suatu proses pengenalan pembicara dari informasi yang terkandung dalam gelombang suara yang diinputkan (Utami dkk., 2015). *Speaker recognition* dibagi menjadi dua bagian, yaitu *speaker verification* dan *speaker identification* (Utami dkk., 2015).

Speaker verification adalah proses pemeriksaan ulang seorang pembicara, sehingga perlu diketahui terlebih dahulu identitas pembicara tersebut berdasarkan data yang telah dimasukkan (Utami dkk., 2015).

Speaker identification adalah proses untuk mencari dan mendapatkan identitas dari seorang pembicara dengan membandingkan fitur suara yang dimasukkan dengan semua fitur suara pembicara yang ada di dalam *database* (Utami dkk., 2015).

2.2 MFCC (Mel-Frequency Cepstrum Coefficient)

MFCC didasarkan atas variasi *bandwidth* kritis terhadap frekuensi pada telinga manusia yang merupakan filter yang bekerja secara linier pada frekuensi rendah dan bekerja secara logaritmik pada frekuensi tinggi (Sanjaya, 2014). Filter ini digunakan untuk menangkap karakteristik fonetik penting dari sinyal ucapan. Untuk meniru kondisi telinga, karakteristik ini digambarkan dalam skala mel-frekuensi, yang merupakan frekuensi linier di bawah 1000 Hz dan frekuensi logaritmik di atas 1000 Hz (Fadli, 2016). Biasanya frekuensi pencuplikan yang digunakan di atas 10000 Hz agar dapat meminimalkan efek aliasing pada konversi analog-digital (Fadli, 2016). Algoritma MFCC diawali dengan *frame blocking* sinyal suara, *windowing*, FFT, *Mel-Frequency Wrapping*, dan *Cepstrum* (Utami dkk., 2015). Hasil akhir dari proses ini adalah *cepstrum coefficient* dalam bentuk *mel-frequency* yang merepresentasikan ciri sinyal suara (Fadli, 2016).

2.2.1 Frame Blocking

Dalam langkah ini sinyal wicara kontinyu diblok menjadi *frame-frame* N sampel. *Frame* pertama terdiri dari N sampel pertama. *Frame* kedua dengan M sampel setelah *frame* pertama, dan *overlap* dengan N–M sampel. Dengan cara yang sama, *frame* ketiga dimulai 2M sample setelah *frame* pertama (atau M sampel setelah *frame* kedua) dan *overlap* dengan N–2M sampel. Proses ini berlanjut hingga semua wicara dihitung dalam satu atau banyak *frame*. Proses *framing* ini dilakukan terus sampai seluruh sinyal dapat terproses. Selain itu, proses ini umumnya dilakukan secara *overlapping* untuk setiap *frame*-nya. Panjang daerah *overlap* yang umum digunakan adalah kurang lebih 30% sampai 50% dari panjang *frame*.

2.2.2 Windowing

Proses *windowing* ini dilakukan pada setiap *frame* hasil dari proses *framing*. Hal ini dilakukan untuk mengurangi terjadinya kebocoran *spectral* atau *aliasing*. Rumus berikut menunjukkan fungsi *window* terhadap sinyal suara yang diinputkan.

$$x(n) = xi(n) * w(n) \quad ...(2.1)$$

Keterangan :

n : Urutan Nilai Sampel

$x(n)$: Nilai Sampel Hasil Windowing

$xi(n)$: Nilai Sampel dari *Frame* Sinyal ke i

i : Urutan *Frame*

$w(n)$: Fungsi *window*

Fungsi *window* yang paling sering digunakan dalam aplikasi *speaker recognition* adalah *Hamming Window*. Berikut persamaan dari *Hamming Window*.

$$w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N-1}) \quad ...(2.2)$$

Keterangan :

n : Urutan Nilai Sampel

N : Jumlah Sampel dalam 1 *Frame*

2.2.3 Fast Fourier Transform (FFT)

Setiap *frame* dikonversi dari domain waktu ke domain frekuensi untuk mendapatkan spectrum frekuensinya. Hal ini dilakukan untuk mempermudah komputasi dan analisa. Metode yang dilakukan untuk mendapatkan spectrum frekuensi adalah *Fast Fourier Transform*

Berikut adalah rumus untuk perhitungan FFT.

$$x_n = \sum_{k=0}^{N-1} x_k e^{-2\pi ikn/N} \quad ...(2.3)$$

Keterangan :

x_k : Nilai Hasil *Windowing* urutan ke k

N : Jumlah Sampel dalam 1 *Frame*

n : Urutan Nilai Sampel

i : Bilangan Imajiner ($\sqrt{-1}$)

2.2.4 Mel Frequency Wrapping

Mel Frequency Wrapping umumnya dilakukan dengan menggunakan *filterbank*. *Filterbank* adalah salah satu bentuk filter yang dilakukan dengan tujuan untuk mengetahui ukuran dari band frekuensi tertentu dalam sinyal suara, pada MFCC digunakan *Mel Filterbank* yang menggunakan skala mel. Skala mel bersifat linear pada frekuensi dibawah 1000 Hz, dan logaritmik pada frekuensi diatas 1000 Hz (Utami dkk., 2015). Untuk mencari skala mel, rumus yang digunakan adalah sebagai berikut.

$$mel(f) = 2595 * \log_{10}(1 + \frac{f}{700}) \quad ...(2.4)$$

Keterangan:

f : nilai hasil FFT

2.2.5 Discrete Cosine Transform

Langkah selanjutnya yaitu mengubah spektrum log mel menjadi domain waktu. Hasil ini disebut *mel frequency cepstrum coefficient* (MFCC). Representasi *cepstral* dari *spectrum* wicara memberikan representasi baik dari sifat-sifat *spectral* lokal sinyal untuk analisis *frame* yang diketahui. Karena koefisien *mel spectrum* adalah bilangan nyata. Dengan mengubahnya menjadi domain waktu menggunakan *discrete cosine transform* (DCT). MFCC (C_n) dapat dihitung dengan rumus sebagai berikut.

$$C_n = \sum_{k=0}^{N-1} (\log S_k) \cos\left[n(k - \frac{1}{2}) \frac{\pi}{N} \right] \quad ...(2.5)$$

Keterangan :

S_k : Hasil *Mel Frequency Wrapping*

n : Urutan Nilai Sampel

N : Jumlah Sampel dalam 1 *Frame*

2.3 Hidden Markov Model

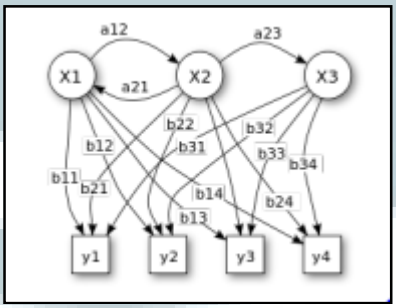
HMM (*Hidden Markov Model)* merupakan model statistik dimana suatu sistem yang dimodelkan diasumsikan sebagai *markov process* dengan kondisi yang tidak terobservasi (Prasetyo, 2010). Suatu HMM dapat dianggap sebagai jaringan Bayesian dinamis sederhana.

Dalam *Markov Model* biasa, setiap keadaan dapat terlihat langsung oleh pengamat. Oleh karena itu, kemungkinan dari transisi antar kondisi menjadi satu-satunya parameter teramati. Dalam HMM, keadaan tidak terlihat secara langsung. Tetapi *output* yang bergantung terhadap keadaan tersebut terlihat. Setiap kondisi

memiliki distribusi kemungkinan disetiap *output* yang mungkin. Oleh karena itu, urutan langkah yang dibuat oleh HMM memberikan suatu informasi tentang urutan dari keadaan. Perlu dipahami, bahwa sifat *hidden* "tersembunyi" menunjuk pada kondisi langkah yang dilewati model, bukan kepada parameter-parameter dari model tersebut. Walaupun parameter model diketahui, model tetap tersembunyi.

HMM adalah variasi dari *finite state machine* yang memiliki kondisi tersembunyi Q, suatu nilai *output* O (observasi), kemungkinan transisi A, kemungkinan *output* B, sebuah kondisi awal Π.

Gambar 2.1 di bawah menjelaskan bagaimana cara kerja dari *Hidden Markov Model*.



Gambar 2.1 *Hidden Markov Model* (Prasetyo, 2010)

Keterangan:

x : Kondisi

y : *Output* yang mungkin

a : Kemungkinan keadaan transisi dari *state* awal ke tujuan

b : Kemungkinan *output*

Permasalahan yang dapat diselesaikan dengan HMM adalah sebagai berikut.

2.3.1 Training

Pengertian dari operasi *training* dalam HMM adalah perhitungan probabilitas dari urutan nilai observasi yang diberikan oleh HMM. Nilai probabilitas pada setiap langkah observasi digunakan dalam perhitungan selanjutnya. Masalah ini dapat diselesaikan dengan Algoritma *Forward* dan *Backward*. Algoritma *Forward* memiliki langkah sebagai berikut (Gultom dkk., 2014).

a. Inisialisasi

Inisialisasi pada Algoritma *Forward* dapat dilakukan dengan rumus sebagai berikut.

$$\alpha_1(i) = \Pi_i b_i(O_1) \quad ...(2.6)$$

Keterangan:

α : persentase *forward*

i : Titik tujuan

Π : Kondisi Awal

b : Persentase Output

b. Induksi

Induksi pada Algoritma *Forward* dapat dilakukan dengan rumus sebagai berikut.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \quad ...(2.7)$$

Keterangan :

N : jumlah kondisi

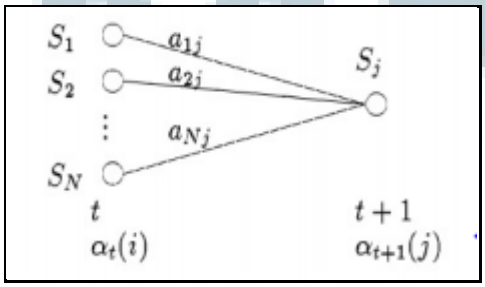
a_{ij} : Probabilitas transisi dari *state* i ke *state* j

c. Terminasi

Terminasi pada Algoritma *Forward* dapat dilakukan dengan rumus sebagai berikut.

$$P(O \mid A, B, \Pi) = \sum_{i=1}^{N} \alpha_t(i) \quad ...(2.8)$$

Ilustrasi pada Algoritma *Forward* dapat dilihat seperti gambar 2.2, dimana S merupakan *state*, a merupakan probabilitas transisi, dan t adalah waktu menuju *state* tujuan.



Gambar 2.2 Algoritma *Forward* (Gultom dkk., 2014)

Algoritma *Backward* memiliki langkah sebagai berikut (Gultom dkk., 2014).

a. Inisialisasi

Inisialisasi pada Algoritma *Backward* dapat dilakukan dengan rumus sebagai berikut.

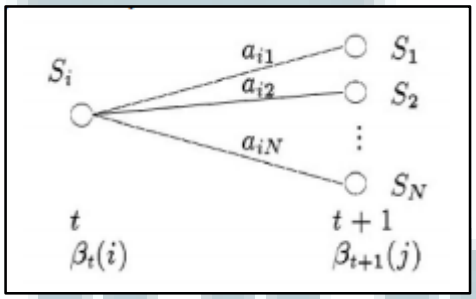
$$\beta_T(i) = 1 \qquad \text{...(2.9)}$$

b. Induksi

Induksi pada Algoritma *Backward* dapat dilakukan dengan rumus sebagai berikut.

$$\beta_t(i) = \sum_{i=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad \text{...(2.10)}$$

Ilustrasi pada Algoritma *Backward* dapat dilihat seperti Gambar 2.3, dimana S merupakan *state*, a merupakan probabilitas transisi, dan t adalah waktu menuju *state* tujuan.



Gambar 2.3 Algoritma *Backward* (Gultom dkk., 2014)

2.3.2 Inferensi

Pengertian dari operasi inferensi dalam HMM adalah penarikan kesimpulan berdasarkan asumsi yang diperoleh dari nilai probabilitas observasi yang didapat sebelumnya pada operasi evaluasi. Operasi ini juga sering kali digunakan untuk mencari nilai optimum. Masalah ini dapat diselesaikan dengan Algoritma *Forward*.

2.4 Akurasi

Akurasi adalah besarnya hasil ketepatan dalam suatu populasi (Zhu dkk., 2010). Persentase akurasi dapat dilihat dari jumlah ketepatan respon pada *voice recognition* (Iqbal dkk., 2011). Ketepatan respon pada *voice recognition* dibagi menjadi *true acceptance* dan *true rejection*. *True acceptance* atau *true positive* adalah ketepatan respon positif yang diberi label benar oleh sistem (Han, 2011). *True rejection* atau *true negative* adalah ketepatan respon negatif yang diberi label salah oleh sistem (Han, 2011). Pada perhitungan persentase tingkat akurasi, dapat dihitung dengan menambahkan *true acceptance* dan *true rejection* lalu dibagi dengan total pengujian lalu dikalikan dengan 100% (Iqbal dkk., 2011).