BAB 2

LANDASAN TEORI

2.1 Pengenalan Ucapan

Pengenalan ucapan atau suara(*speech recognition*) adalah suatu teknik yang memungkinkan sistem komputer untuk menerima *input* berupa kata yang diucapkan. Kata-kata tersebut diubah bentuknya menjadi sinyal *digital* dengan cara mengubah gelombang suara menjadi sekumpulan angka lalu disesuaikan dengan kode-kode tertentu dan dicocokkan dengan suatu pola yang tersimpan dalam suatu perangkat. Hasil dari identifikasi kata yang diucapkan dapat ditampilkan dalam bentuk tulisan atau dapat dibaca oleh perangkat teknologi.

Ukuran kosakata (*vocabulary*) dari sistem pengenalan suara mempengaruhi kompleksitas, parameter pelatihan dan akurasi sistem.Beberapa aplikasi pengenalan suara hanya memerlukan beberapa kata, sedangkan yang lainnya memerlukan kamus yang sangat besar (misalnya mesin pendiktean). Terdapat 4 jenis ukuran kosakata, yaitu:

a. Kosakata ukuran kecil *(small vocabulary)* yang terdiri dari puluhan kata.
b. Kosakata ukuran sedang *(medium vocabulary)* yang terdiri dari ratusan kata.
c. Kosakata ukuran besar*(large vocabulary)* yang terdiri dari ribuan kata,
d. Kosakata ukuran sangat besar *(very large vocabulary)*yang terdiri dari puluhan ribu kata.

   Sumber: (http://www.speech.cs.cmu.edu/comp.speech/Section6/Q6.1.html)

Teknologi pengenalan suara berkembang dengan begitu pesat. Beberapa jenis diantaranya yaitu:

- Sintesis suara *(speech synthesis)*: dari teks ke suara.
- Pengenalan pembicara *(speaker recognition)*: dari suara ke identitas pembicara.
- Pendiktean *(dictation)*: dari suara ke teks.
- Rangkuman suara *(speech summarization)*: dari suara ke teks sederhana.
- Pengkategorian suara *(speech categorization)*: dari suara ke label kelas.
- Pengertian suara *(speech understanding)*: dari suara ke representasi makna suara.
- Pemrosesan dialog *(dialog processing)*: dari suara ke makna suara yang interaktif.
- Penerjemah suara *(speech translation)*: dari suara ke suara dalam bahasa lain.

Sumber: (http://web.iti.upv.es/~evidal/students/doct/ta/transp/t4mtS2S2p.pdf).

Kegunaan aplikasi pengenal ucapan yaitu:

- Pada bidang komunikasi:
    a. Sebagai komando suara pada komputer untuk melakukan perintah seperti membuka, menyimpan, menutup *file* dan sebagainya.
    b. Sebagai alat pendiktean untuk membuat laporan dimana aplikasi akan menuliskan teks sesuai dengan yang diucapkan pembicara.
- Pada bidang kesehatan, alat pengenal ucapan digunakan untuk membantu para penyandang cacat agar dapat beraktivitas dengan memerintahkan alat-alat bantu melalui suaranya.

- Pada bidang militer, aplikasi alat pengenal ucapan digunakan pada pengatur lalu lintas udara yang dikenal dengan *Air Traffic Controllers* (ATC). Alat ini dipakai untuk memberi keterangan mengenai keadaan lalu lintas udara seperti radar, cuaca, dan navigasi dengan kata lain sebagai pengganti operator yang memberikan informasi kepada pilot dengan cara berdialog.
- Secara umum, pengenal ucapan banyak digunakan karena mudah yakni hanya menggunakan suara dan cepat prosesnya.

Kekurangan pengenalan ucapan yaitu:

- Rawan terhadap gangguan sinyal suara lain terutama di tempat yang ramai.
- Kata-kata yang diucapkan dapat sulit dikenali karena cara pengucapan yang berbeda walaupun oleh pembicara yang sama. Intonasi, logat, dan kecepatan pengucapan sangat mempengaruhi.
- Bahasa lisan seringkali diucapkan tidak sesuai dengan kaidah tata bahasa yang baku.
- Jumlah kata yang dapat dikenali masih terbatas.

  Sumber: (http://www.faqs.org/docs/Linux-HOWTO/Speech-Recognition-HOWTO.html)

2.2 Pocketsphinx

Pocketsphinx merupakan *library* pengenalan ucapanversi *mobile application* dari sistem *Sphinx*yang dirancang oleh *Carnegie Mellon University*.Metode yang digunakan dalam sistem *speech recognitionPocketphinx* ini yaitu metode *Hidden Markov Model*. Proses pembelajaran unit-unit suara disebut *training*, sedangkan

proses menggunakan pengetahuan yang diperoleh untuk menyimpulkan urutan yang paling mungkin dari unit dalam sinyal yang diberikan disebut *decoding*, atau secara sederhana disebut pengenalan (*recognition*). Karena terdapat dua proses tersebut maka diperlukan *SPHINX trainer* dan *SPHINX decoder*.

2.2.1 Komponen Untuk Training

Sphinx trainer terdiri dari kumpulan program yang setiap bagiannya memiliki tugas yang spesifik.Kita harus mempelajari parameter-parameter dari model unit suara dengan menggunakan kumpulan sampel sinyal suara pada *training database*.*Transcript file* adalah *file* yang berisi urutan kata-kata dan suara yang tidak diucapkan yang ditulis bersesuaian dengan sinyal suara. Dibutuhkan juga kamus (*dictionary*) yang memetakan setiap kata ke urutan unit suara.Jadi, sebagai tambahan pada sinyal suara, harus ada sekumpulan transkrip untuk *database* (dalam sebuah file tunggal) dan dua kamus.Satu kamus dimana kata-kata yang sah dalam bahasa dipetakan pada urutan unit suara (atau unit sub-kata) dan satu lagi dimana suara non-ucap dipetakan ke *non-speech* atau suara seperti unit suara.Bentuk yang pertama adalah kamus bahasa (*language dictionary*) dan yang kedua yaitu pengisi kamus (*filler dictionary*).

Sumber: (http://www.speech.cs.cmu.edu/sphinx/tutorial.html)

Secara ringkas, komponen untuk training yaitu:

1. *Phonetic dictionary* (kamus fonetik) dengan *extention* .dic
2. *Phoneset file*dengan *extention* .phone
3. *Transcript file* dengan *extention* .transcription
4. *Language model*(model bahasa) dengan *extention* .DMP

5. Daftar *file* untuk *training* dengan *extention* .fileids
6. *Filler dictionary* dengan *extention* .fillers
7. *File* suara dengan format wav 16 kHz, 16 bit, mono.

Sumber: (http://cmusphinx.sourceforge.net/wiki/tutorialam)

2.2.2 Komponen Untuk Decoding

Decoder juga terdiri dari satu set program, yang telah disusun untuk memberikan *executable* tunggal yang akan melakukan tugas pengenalan saat diberi *input* yang tepat. *Input* yang perlu diberikan adalah model akustik yang terlatih, sebuah *file* indeks model, model bahasa, kamus bahasa, kamus pengisi, dan kumpulan sinyal akustik yang perlu dikenali. Data yang akan dikenali secara umum disebut sebagai data uji (*test data*).

Secara ringkas, komponen yang diperlukan untuk proses *decoding* yaitu:

1. *Source code decoder*
2. *Language dictionary*
3. *Filler dictionary*
4. *Language model*
5. Data uji

2.2.3 Penulisan Fonetik

Dalam file transkrip misalnya terdapat tiga kalimat sebagai berikut:

THIS CAR THAT CAT (*file1*)

CAT THAT RAT (*file2*)

THESE STARS (*file3*)

dan kamus bahasa dari kata-kata di atas yaitu:

CAT	K	AE	T		
CAR	K	AA	R		
RAT	R	AE	T		
STARS	S	T	AA	R	S
THIS	DH	I	S		
THAT	DH	AE	T		
THESE	DH	IY	Z		

maka frekuensi terjadinya untuk setiap fonetik adalah sebagai berikut (dalam skenario nyata di mana harusmelatih model *triphone*, maka harus menghitung *triphones* juga):

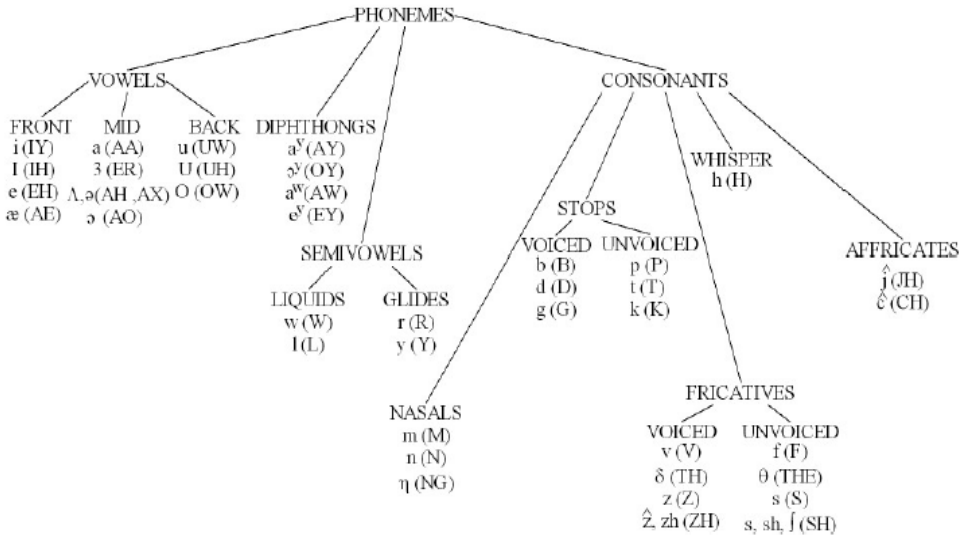
K	3	S	3
AE	5	IY	1
T	6	I	1
AA	2	DH	4
R	3	Z	1

sumber: http://www.speech.cs.cmu.edu/sphinx/tutorial.html

Penulisan fonetik pada *pocketsphinx* menggunakan sistem *Arpabet*.*Arpabet* adalah transkripsi fonetik yang dikembangkan oleh *Advanced Research Projects Agency* (ARPA) sebagai bagian dari *Speech Understanding Project* (1971-1976).*Arpabet* mewakili setiap fonem dari dalam bahasa Inggris Amerika umum (*General American English*) dengan urutan yang berbeda dari karakter ASCII.

Dalam *Arpabet*, setiap fonem diwakili oleh satu atau dua huruf. Digit digunakan *indicatorstress* (penekanan) dan ditempatkan pada akhir suku kata vokal yang ditekankan.Daftar *Arpabet* dapat dilihat pada Lampiran 1.

Sumber: (http://www.speech.cs.cmu.edu/cgi-bin/cmudict).



Gambar 2.1Fonetik Bahasa Inggris aksen Amerika
Sumber: (Lundsgaard 2006, 12)

2.3 Hidden Markov Model

Riset mengenai pengenalan pola berkembang begitu pesat. Lahirnya metode *Hidden Markov* menjadi salah satu pemicu pembaruan teknik pengolahan *speech*(ucapan / suara) yang dapat menghasilkan tingkat kesalahan seminimal mungkin dan meningkatkan ketangguhan pengklasifikasian pola pada berbagai kondisi yang kurang baik.

Hidden Markov Model (HMM) adalah suatu model statistik dari sebuah sistem yang diasumsikan sebuah proses *Markov* dengan parameter yang tak diketahui. Kita harus menentukan parameter-parameter tersembunyi (*state*) dari parameter-parameter yang dapat diamati. Parameter-parameter yang ditentukan kemudian dapat digunakan untuk analisis yang lebih jauh, misalnya untuk aplikasi *pattern recognition*.

Dalam HMM, *state* tidak dapat diamati secara langsung, akan tetapi yang dapat diamati adalah variabel-variabel yang terpengaruh oleh *state*. Setiap *state* memiliki distribusi probabilitas atas *token-token output* yang mungkin muncul. Oleh karena itu rangkaian *token* yang dihasilkan oleh HMM memberikan sebagian informasi tentang sekuens *state-state*.

Transisi pada rantai *Markov* yaitu:

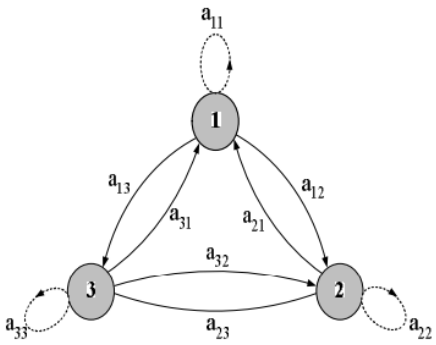
- Transisi dari suatu keadaan tergantung pada keadaan sebelumnya.

$$P[q_t = j | q_{t-1} = i,\ q_{t-2} = k.....] = P[q_t = j \mid q_{t-1} = i]$$

- Transisi keadaan bebas terhadap waktu.

$$a_{ij} = P[q_t = j \mid q_{t-1} = i\ ]$$

Berikut ini adalah contoh gambar rantai *Markov*.



Gambar 2.2 Rantai Markov
Sumber: (Ricky, 2004)

Elemen pada HMM yaitu:

1. N, jumlah keadaan (*state*) dalam model.
2. M, jumlah simbol observasi yang berbeda tiap keadaan.

3. Distribusi keadaan transisi A = $\{a_{ij}\}$ dengan

$$a_{ij} = P[q_{t+1} = j \mid q_t = j], \qquad 1 \le i, j \le N$$

4. Distribusi probabilitas simbol observasi B = $\{b_j(k)\}$

dengan $b_j(k) = P(o_t = v_k \mid q_t = j), 1 \le J \le N, 1 \le k \le M$

5. Distribusi keadaan awal $\pi = \{\pi_i\}$

$$\pi_i = P[q_t = i] \qquad 1 \le i \le N$$

Speech recognition (pengenalan ucapan) merupakan pengenalan sebuah kata berurutan dari ucapan manusia. Otak manusia berfungsi sebagai generator teks yang menghasilkan kumpulan kata *W*. Kumpulan kata berasal dari saluran akustik yang terdiri dari artikulasi pembicara dan proses akustik lainnya yang mengkonversi kumpulan teks ke dalam suatu bentuk gelombang akustik yang dapat didengar. Saluran akustik dalam komunikasi verbal bertindak sebagai *data transducer* dan komposer.*Speech recognizer* adalah dekoder yang menampilkan proses pembalikkan dari pesan ke bentuk gelombang suara. Oleh karena itu, dekoder menunjukkan keputusan akhir maksimum dari pemeriksaan urutan kata $\hat{W}$ seperti berikut:

$$\widehat{W} = \underset{w}{^{argmax}} P(W \mid X) = \underset{w}{^{argmax}} P(X \mid W)P(W) \qquad (2.1)$$

dimana P(X | W) adalah nilai dari pemodelan akustik dan P(W) adalah nilai dari model bahasa.

Sistem pengenalan ucapan terdiri dari beberapa komponen dasar, diantaranya:

- *Acoustic feature extraction*: berfungsi untuk mengekstraksi pengenalan suara dari gelombang suara. Biasanya termasuk analisis *cepstral*jangka

pendek yang mengolah suatu fitur *vector* dari frekuensi rendah (10-16) koefisien *cepstral* untuk setiap 10ms. $X = (x_1, \dots, x_T)$ untuk merepresentasikan fitur observasi akustik urutan *vector*

- *Acoustic modeling*: berfungsi untuk menyediakan pemodelan statistik bagi observasi akustik *X*.
- *Language modeling*: berfungsi untuk menyediakan linguistik dan batasan *grammar* (tata bahasa) untuk urutan teks *W*. *Language model*(model bahasa) seringkali dibuat berdasarkan *N-gram language model*. Sebuah N-gram model bahasa dibentuk dari $P(w_n|w_1, \dots, w_{n-1})$ dimana peluang dari kata yang diamati w_n diberikan deretan kata $w_1, \dots, w_{n-1}$
- *Decoding engine*: berfungsi untuk mencari urutan kata terbaik yang diberikan fitur dan model. Untuk pengenalan suara berdasarkan pemodelan HMM, maka *decoding* menggunakan *Viterbi*. Untuk peluang observasi diskrit, kumpulan kata *W* diperoleh dari:

$$\widehat{W} = \underset{w}{^{argmax}} P(X, \Lambda w_Q | W) \qquad (2.2)$$

dimanaΛw_Q adalah *state* terbaik dari W, X dan model Λ. Untuk densitas yang bekelanjutan,

$$\widehat{W} = \underset{w}{^{argmax}} \log f(X, \Lambda w_Q | W) \qquad (2.3)$$

dimana rumus diatas berdasarkan nilai *log-likelihood* sepanjang urutan *state* terbaik W_Q

Pemodelan HMM adalah pemodelan statistik sinyal suara yang populer dan bagus.Diberikan suatu ucapan, misalkan $X = (x_1, x_2, \dots, x_T)$ menjadi fitur urutan

vektor terekstraksi dari gelombang suara, dimana x_T menunjukkan suatu pengukuran vektor jangka pendek dan secara konvensional sebuah vektor *cepstral*.

Suatu permintaan awal *N-state* rantai *Markov* diperintah oleh suatu transisi *state* peluang matriks $A = [a_{ij}]$, dimana a_{ij} adalah peluang pembuatan transisi dari state i ke state j. Asumsikan bahwa pada $t = 0$, *state* pada sistem q_0 dikhususkan oleh suatu inisial *state* dengan peluang $\pi_i = P(q_0 = i)$. Lalu, untuk setiap urutan $stateq = (q_0, q_1, \dots, q_T)$, peluang q yang dihasilkan dari rantai *Markov* yaitu:

$$P(q \mid A, \pi) = \pi_{q_0} a_{q_0 q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T} \qquad (2.4)$$

Anggap pada sistem, ketika di $stateq_t$, keluarkan observasi x_t menurut distribusi $b_{q_t}(x_t) = P(x_t|q_t), q_t = 1,2, \dots, N$. HMM digunakan sebagai distribusi ungkapan suara X lalu didefinisikan sebagai:

$$P\left(X \mid \pi, A, \{b_j\}_{j=1}^{N}\right) = P(X \mid \Lambda) = \sum_{\mathbf{q}} P(X, \mathbf{q} \mid \Lambda)$$
$$= \sum_{\mathbf{q}} P(X \mid \mathbf{q}, \Lambda)\, P(\mathbf{q} \mid \Lambda) = \sum_{\mathbf{q}} \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1} q_t} b_{q_t}(\mathbf{x}_t) \qquad (2.5)$$

dimana$\Lambda = \left(\pi, A, \{b_j\}_{j=1}^{N}\right)$ adalah himpunan parameter untuk model. Seperti pada persamaan 2.5, $\{b_{q_t}\}$ mendefinisikan distribusi untuk observasi jangka pendek dan A mencirikan sifat dan hubungan di antara *state* yang berbeda dari proses penghasilan suara. Secara umum, N (jumlah total *state*) lebih kecil dari T (waktu selama pengucapan suara). Urutan *state* q menunjukkan derajat stabilitas tertentu diantara $adjacentq\frac{1}{t}s$.

Tiga masalah dasar dalam HMM yang harus dikaji ulang yaitu masalah evaluasi, masalah *decoding* dan masalah estimasi.Masalah evaluasi adalah memperkirakan peluang $P(X|\Lambda)$ dari pengamatan fitur suara urutan vektor X yang diberikan *Hidden Markov Model*. Masalah *decoding* yaitu untuk menemukan urutan *state* terbaik q yang optimal dari urutan fitur suara X. Karena *state* dalam HMM terhubung ke kata dan kelas kata, urutan kata dalam pengucapan suara dapat diidentifikasikan dengan menelusuri *label* kata pada urutan state q. Masalah yang ketiga yaitu estimasi yang diperlukan untuk memperkirakan parameter HMM Λ dari himpunan sampel yang sudah dilatih. Pendekatan biasa yaitu berdasarkan prinsip *Maximum Likelihood* (ML) dan himpunan parameter model Λ diestimasi sehingga *likelihood* pada pelatihan data menjadi maksimal.Beragam ML yang efisien tergantung pada algoritma yang dikembangkan dalam pengenalan suara untuk HMM, seperti *Baum-Welch* dan algoritma segmental *k-means*.(Chou, Wu., Biing H. Juang, 2003, p 23-26).

2.3.1 Perhitungan Probabilitas Maju-Mundur (*Forward-Backward*)

Algoritma rekursif yang efisien untuk perhitungan fungsi *likelihood* $f_{X|M}(X|M)$ adalah algoritma maju-mundur. Perhitungan pada metode maju-mundur memanfaatkan struktur yang sangat teratur dan berulang.

Dalam metode ini, variabel αt probabilitas maju (i) didefinisikan sebagai probabilitas gabungan dari urutan observasi parsial X = [x (0), x (1), ..., x(t)] dan *statei* pada waktu *t*, dari model M:

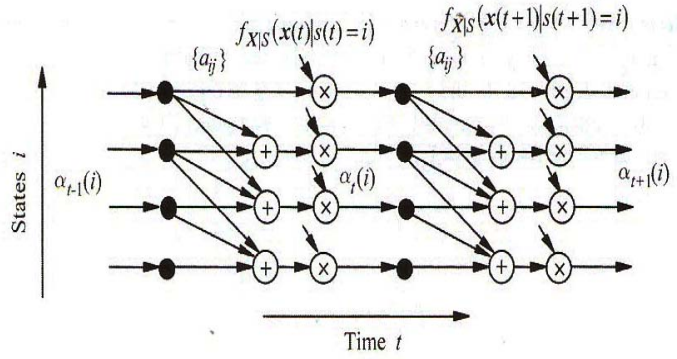
$$\alpha_t(i) = f_{X,S|\mathcal{M}}\left(\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(t), s(t) = i \,|\, \mathcal{M}\right) \tag{2.6}$$

Variabel probabilitas maju $\alpha_t(i)$ pada persamaan (2.6) dapat dinyatakan dalam bentuk rekursif pada waktu *t*-1, $\alpha_{t-1}(i)$:

$$\begin{aligned}
\alpha_t(i) &= f_{X,S|\mathcal{M}}\left(\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(t), s(t) = i \,|\, \mathcal{M}\right) \\
&= \left(\sum_{j=1}^{N} f_{X,S|\mathcal{M}}\left(\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(t-1), s(t-1) = j \,|\, \mathcal{M}\right) a_{ji}\right) \\
&\quad f_{X|S,\mathcal{M}}\left(\boldsymbol{x}(t) \,|\, s(t) = i, \mathcal{M}\right) \\
&= \sum_{j=1}^{N} \left(\alpha_{t-1}(j) a_{ji}\right) f_{X|S,\mathcal{M}}\left(\boldsymbol{x}(t) \,|\, s(t) = i, \mathcal{M}\right)
\end{aligned} \tag{2.7}$$

Gambar 2.3Ilustrasi suatu jaringan untuk perhitungan probabilitas maju pada HMM empat *state* kiri-kanan. Kemungkinan (*likelihood*) dari urutan observasi X=[x(0), x(1), …, x(T-1)] yang diberi model M dapat dinyatakan dalam bentuk probabilitas maju sebagai:

$$\begin{aligned}
f_{X|\mathcal{M}}\left(\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(T-1) \,|\, \mathcal{M}\right) &= \sum_{i=1}^{N} f_{X,S|\mathcal{M}}\left(\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(T-1), s(T-1) = i \,|\, \mathcal{M}\right) \\
&= \sum_{i=1}^{N} \alpha_{T-1}(i)
\end{aligned} \tag{2.8}$$



Gambar 2.3Jaringan perhitungan probabilitas maju HMMkiri-kanan
Sumber: (Vaseghi 2008, p163)

Sama halnya dengan definisi konsep peluang maju, sebuah peluang mundur didefinisikan sebagai peluang *statei* pada waktu *t* yang diikuti oleh urutan observasi parsial [x(t+1), x(t+2), …., x(T-1)]

$$\begin{aligned}\beta_t(i) &= f_{X,S|\mathcal{M}}\left(s(t)=i, \boldsymbol{x}(t+1), \boldsymbol{x}(t+2), \ldots, \boldsymbol{x}(T-1)\,|\,\mathcal{M}\right) \\ &= \sum_{j=1}^{N} a_{ij} f_{X,S|\mathcal{M}}\left(s(t+1)=j, \boldsymbol{x}(t+2), \boldsymbol{x}(t+3), \ldots, \boldsymbol{x}(T-1)\,|\,\mathcal{M}\right) \\ &\quad \times f_{X|S,\mathcal{M}}\left(x(t+1)\,|\,s(t+1)=j, \mathcal{M}\right) \\ &= \sum_{j=1}^{N} a_{ij}\beta_{t+1}(j) f_{X|S,\mathcal{M}}\left(\boldsymbol{x}(t+1)\,|\,s(t+1)=j, \mathcal{M}\right)\end{aligned} \tag{2.9}$$

(Vaseghi 2008, p156-157)

2.3.2 Model Re-Estimasi Baum-Welch

Masalah dalam pelatihan HMM adalah mengestimasi model parameter M = (π, A, F) untuk himpunan *X* yang diberikan. Parameter-parameter ini adalah *state* awal pada peluang π, parameter-parameternya adalah peluang *state* awal π, peluang *state* transisi matriks *A* dan densitas kontinu (atau diskrit) observasi*statepdf* (*probability density function*). Parameter HMM diperkirakan dari sekumpulan contoh-contoh training {X = [x(0), ..., x (T-1)]}, dengan memaksimalkan $f_{x|M}$ (X|M), *likelihood* dari model dan data pelatihan. Metode *Baum-Welch* adalah iterasi dari metode *likelihood* yang dimaksimalkan berdasarkan probabilitas maju-mundur. Untuk HMM

sebuah M, probabilitas *posterior* dari transisi pada waktu *t* dari *statei* ke *statej* dari model M, diberikan sebuah urutan observasi berupa X.

$$\begin{aligned}\gamma_t(i,j) &= P_{S|X,\mathcal{M}}(s(t)=i, s(t+1)=j \,|\boldsymbol{X}, \mathcal{M}) \\ &= \frac{f_{S,X|\mathcal{M}}(s(t)=i, s(t+1)=j, \boldsymbol{X}\,|\mathcal{M})}{f_{X|\mathcal{M}}(\boldsymbol{X}\,|\mathcal{M})} \\ &= \frac{\alpha_t(i)a_{ij}f_{X|S,\mathcal{M}}(\boldsymbol{x}(t+1)\,|s(t+1)=j, \mathcal{M})\beta_{t+1}(j)}{\sum_{i=1}^{N}\alpha_{T-1}(i)}\end{aligned} \tag{2.10}$$

dimana $f_{s,x}|M_{(s(t)}=i$, $s(t+1)=j$, $X|M)$ adalah *pdf* join dari state $s(t)$ dan $s(t+1)$ dan urutan observasi X, dan $f_{x|s}$ $(x(t+1)|s(t+1)=i)$ adalah *state* observasi *pdf* untuk *statei*. Untuk sebuah densitas observasi diskrit HMM, persamaan 2.10 diganti dengan *state* observasi *pmf (probability mass function)*diskrit $Px|s$ $(x+1)|s(t+1)=i)$. Probabilitas *posterior* dari *statei* pada waktu *t* dengan model *M* dan observasi *X* adalah

$$\begin{aligned}\gamma_t(i) &= P_{S|X,\mathcal{M}}(s(t)=i\,|\boldsymbol{X}, \mathcal{M}) \\ &= \frac{f_{S,X|\mathcal{M}}(s(t)=i, \boldsymbol{X}\,|\mathcal{M})}{f_{X|\mathcal{M}}(\boldsymbol{X}\,|\mathcal{M})} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_{T-1}(j)}\end{aligned} \tag{2.11}$$

Peluang *state* transisi a_{ij} dapat ditulis sebagai berikut

$$a_{ij} = \frac{\text{Expected number of transitions from state } i \text{ to state } j}{\text{Expected number of transitions from state } i} \tag{2.12}$$

daripersamaan (2.10)-(2.12) peluang *state* transisi dapat diperkirakan ulang dengan rasio

$$\overline{a}_{ij} = \frac{\sum_{t=0}^{T-2} \gamma_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)} \tag{2.13}$$

untuk sebuah urutan observasi [x(0), …., x(T-1)] dari panjang *T*, transisi terakhir terjadi pada waktu *T-2* yang diindikasikan sebagai batas teratas dari penjumlahan pada persamaan 2.13. Peluang inisial *state* diestimasikan sebagai berikut.

$$\overline{\pi}_i = \gamma_0(i) \tag{2.14}$$

(Vaseghi 2008, p157-158)

2.3.3 Algoritma Decoding Viterbi

Decoding sinyal berfungsi untuk mendapatkan perkiraan *posterior* maksimum (MAP)dari urutan *state* yang mendasarinya. Urutan *state* MAP dari model *M* diberikan sebuah sinyal observasi X=[x(0), …., x(T-1)] yang menghasilkan :

$$\begin{aligned} s^{MAP} &= \arg\max_s f_{X,s|\mathcal{M}}(X, s|\mathcal{M}) \\ &= \arg\max_s \left(f_{X|s,\mathcal{M}}(X|s,\mathcal{M}) P_{s|\mathcal{M}}(s|\mathcal{M}) \right) \end{aligned}$$

δ_t (*i*) mencatat peluang kumulatif dari jalan terbaik untuk *state i* pada waktu *t*. ψt (*i*) mencatat urutan state terbaik untuk *state i* pada waktu *t*.

Langkah ke-1: Inisialisasi, waktu t=0, atau *state*i=1, …, N

$$\delta_0(i) = \pi_i f_i(x(0))$$

$$\psi_0(i) = 0$$

Langkah ke-2: Perhitungan rekursif urutan *state ML* dan peluang

Untuk waktu t=1, …, T-1

Untuk states i=1, …, N

$$\delta_t(i) = \max [\delta_{t-1}(j) a_{ji}] f_i(x(t))$$

$$\psi_t(i) = \arg\max [\delta_{t-1}(j) a_{ji}]$$

Langkah ke-3: Pemutusan, mengambil *state* akhir yang paling mungkin

$$s^{MAP}(T\text{-}1) = \arg\max_i [\delta_{T-1}(i)]$$

$$Prob_{max} = \max_i [\delta_{T-1}(i)]$$

Langkah ke-4: *Backtracking* melalui urutan *state* yang paling mungkin

Untuk $t = T$-2, …, 0

$$s^{MAP}(t) = \psi_{t+1}[s^{MAP}(t+1)]$$

(Vaseghi 2008, p162-164)

2.4 Pemodelan Bahasa

Pemodelan bahasa (*language modeling*) menangkap aturan dalam bahasa lisan dan digunakan dalam pengenalan suara untuk memperkirakan probabilitas urutan kata.Sementara kendala gramatikal yang dijelaskan dengan tata bahasa bebas konteks (*context free grammars*) telah digunakan untuk kosakata ukuran kecil dan sedang. Untukkosakata berukuran besar seperti *Large Vocabulary Continuous Speech Recognition* (LVCSR)biasanya selalu didasarkan pada data mengacu pada pendekatan. Metode statistik yang paling populer adalah model n-gram, yang mencoba untuk menangkap kendala sintaksis dan semantik bahasa dengan memperkirakan frekuensi urutan sejumlah *n* kata. Asumsi yang dibuat untukpeluang dari *string* kata tertentu yaitu $(W = \omega_1, \omega_2, \ldots \omega_k)$ yang dapat diperkirakan oleh dekomposisi urutan maju (*forward sequential decomposition*) berikut ini:

$$P(W) = \prod_{i=1}^{k} \Pr(w_i | w_{i-n+1}, \ldots, w_{i-2}, w_{i-1})$$

Sebagai contoh misalnya terdapat kumpulan kata W = $\{w_1,w_2,w_3,w_4\}$, maka peluang P(W) yaitu:

$$P(W) = P(w_2 | w_1).P(w_3 | w_2).P(w_4 | w_3)$$

Syarat awal untuk memperkirakan model bahasa n-gram adalah ketersediaan *corpora*teks (istilah linguistik untuk sekumpulan teks yang banyak dan terstruktur) yang sesuaiuntuk diproses.Model bahasa biasanya diperkirakan dari transkripsi manual *speech corpora* dan dari normalisasi *corpora* teks. Untuk memastikan model yang akurat, teks harus mewakili input audio yang akan ditranskripsi. Model bahasa umumnya dioptimalkan dan dibandingkan dengan mengukur

ambiguitaskumpulan data yang ditinggalkan, disebut sebagai data pembangunan model bahasa.Pengujian ini disebut tes kumpulankeambiguandari model bahasa yang didefinisikan sebagai:

$$\mathrm{Px}(T|M) = P(T|M)^{-\frac{1}{L}} \simeq (\prod_{i=1}^{L} P(w_i|w_{i-2}, w_{i-1}))^{-\frac{1}{L}}$$

untuk teks yang diberikan T = *(ω_1, …. ω_L)*dan model bahasa trigram (misalnya*n* = 3). P (T|M) menunjukkan estimasi model bahasa dari probabilitas teks. Ambiguitas tergantung pada bahasa yang dimodelkan dan model bahasa itu sendiri, yaitu, memberikan perkiraan gabungan dari seberapa baik model dan seberapa kompleksnya bahasa. (Chou, Wu., Biing H. Juang, 2003, p165).

2.5 N-gram

N-gram adalah *substring* sepanjang *n* karakter dari suatu*string*. *N-gram* merupakan sebuah metode yang diaplikasikan untuk pembangkitan kata atau karakter. Metode *n-gram* ini digunakan untuk mengambil potongan-potongan karakter huruf sejumlah n dari sebuah kata atau kalimat yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen.Dokumen akan dibaca kata per kata, dan untuk setiap kata akan dibuat *n-gram*-nya.Sumber: (http://digilib.ittelkom.ac.id/index.php?option=com_content&view=article&id=531:metode-n-gram&catid=20:informatika&Itemid=14).

Di bawah ini adalah contoh n-gram, dengan 2 buah kalimat, yaitu:

This is a test
Thiis is a second test

Dengan massa diskon tetap (*fixed discounted mass*)0.4,model bahasa*trigram* (terbatas pada urutan 3 kata) yang dihasilkan dari korpus teks yang terbatas adalah sebagai berikut:

```
\data\
ngram 1=5
ngram 2=5
ngram 3=4
\1-grams:
-0.8751 This -0.3358
-0.8751 a -0.3010
-0.8751 is -0.3358
-1.1761 second -0.3358
-0.8751 test -0.3979
\2-grams:
-0.2218 This is 0.0000
-0.5229 a second 0.0000
-0.5229 a test. -0.3979
-0.2218 is a 0.0000
-0.2218 second test -0.3979
\3-grams:
-0.2218 This is a
-0.2218 a second test
-0.5229 is a second
-0.5229 is a test
\end\
```

Di sisi kiri dari setiap urutan kata-kata terdapat angka yang merupakan probabilitas, dan di sisi kanan adalah probabilitas *backoff* (kecuali untuk 3-gram).Misalnya pada baris pertama yaitu "-0.8751 This -0.3358". Total banyaknya kata dalam dua kalimat di atas adalah 9. Peluang kata "*This*" dalam korpus teks adalah $\frac{2}{9}$ (kata "*This*" dua kali muncul dari total 9 kata) dikalikan dengan 0.6 dari (1-0.4, massa non-diskon). Dalam model bahasa yang sesuai dengan ARPA maka harus menggunakan *log* sehingga perhitungan menjadi $log\left(\frac{2}{9}(1-0.4)\right) = -0.8751.$

Untuk jumlah kata yang sedikit diperlukan diskon dan peluang *backoff*.Peluang *backoff* terkait dengan ucapan *non-terminated*(ucapan yang belum selesai). Sebagai contoh, jika mengucapkan "This is a" dan kemudian dilanjutkan "test". Tanpa peluang *backoff*, suatu sistem pengenalan ucapan akan gagal mengenali ucapan *non-terminated* karena tidak memiliki probabilitas yang terkait dengan kata sebelumnya.

Cara menghitung probabilitas *backoff* dari "*This*" yaitu:

$$\log\left(\frac{diskon}{1-((total\ kata+1)^{-0.8751})}\right)$$

$$=\log\left(\frac{diskon}{1-((9+1)^{-0.8751})}\right)$$

$$=\log\left(\frac{0.4}{1-0.13332}\right)=-0.3358$$

Ketika menghitung n-gram dengan N > 1, maka perhitungannya yaitu dibagi dengan hitungan (N-1)-gram dimulai dengan urutan kata-kata yang sama. Misalnya, probabilitas yang terkait dengan "*This is*" yaitu log(2/2 * (1 - 0.4)) = -0.2218 karena ada 2 kejadian dari kata "*This*" serta 2 kejadian dari urutan "*This is*".

(Srinivas, K.L., Pranav Jawale, 2011).

2.6 Android

2.6.1 Sistem Operasi Android

Android merupakan sistem operasi berbasis *Linux* untuk telepon seluler (*mobile*) yang menyediakan *platform* terbuka (*open source*) sehingga para pengembang dapat menciptakan aplikasi sendiri.Pada awalnya sistem operasi *Android* dikembangkan oleh *Android Inc.* lalu dibeli oleh *Google*

pada tahun 2005.*Android* berbeda dari sistem operasi *mobile* lainnya yang memiliki kekurangan seperti keterbatasan dari aplikasi pihak ketiga untuk mendapatkan data asli ponsel, keterbatasan berkomunikasi antar proses serta keterbatasan distribusi aplikasi pihak ketiga untuk *platform* mereka. Keunggulan *Android* yaitu setiap aplikasi memiliki tingkatan yang sama antara aplikasi inti dengan aplikasi pihak ketiga. *Application Programming Interface* (API) yang disediakan menawarkan akses ke *hardware*, data-data ponsel atau data sistem sendiri.Pengguna bahkan dapat menghapus aplikasi inti dan menggantinya dengan aplikasi pihak ketiga.

Tabel 2.1 Versi Android

Versi	**API Level**	**Distribusi**
2.0.x / 2.1.x (*Éclair*)	7	24,5 %
2.2.x (*Froyo*)	8	65.9 %
2.3.x (*Gingerbread*)	10	4,0 %
3.x.x (*Honeycomb*)	11	0,3 %

Berdasarkan tabel di atas maka versi *Android* yang paling banyak digunakan adalah *Froyo*.

Empat prinsip pengembangan sistem operasi dan aplikasi *Android* menurut (Hermawan S., 2011),yaitu:

1. Terbuka

   Android dibangun untuk menjadi benar-benar terbuka.Sebagai contoh, sebuah aplikasi dapat mengambil dan mengakses fungsi-fungsi utama ponsel seperti membuat panggilan, mengirim pesan teks, menggunakan kamera.Hal ini memungkinkan para pengembang untuk membuat aplikasi yang lebih baik.

2. Semua aplikasi dibuat sama

   Android tidak membedakan antara aplikasi inti dan aplikasi pihak ketiga, jadi keduanya dapat dibangun dan memiliki akses yang sama ke ponsel.

3. Mendobrak batasan-batasan aplikasi

   Pengembang dapat menggabungkan informasi misalnya dari *website* dengan data individu dari ponsel.Selain itu pengembang juga dapat membuat aplikasi untuk melihat lokasi dan terkoneksi dengan teman-temannya.

4. Pengembangan aplikasi yang cepat dan mudah

   Android menyediakan akses ke berbagai *library* dan *tool* sehingga aplikasi menjadi lebih kaya akan fitur-fitur canggih.

2.6.2 Fitur dan Arsitektur Android

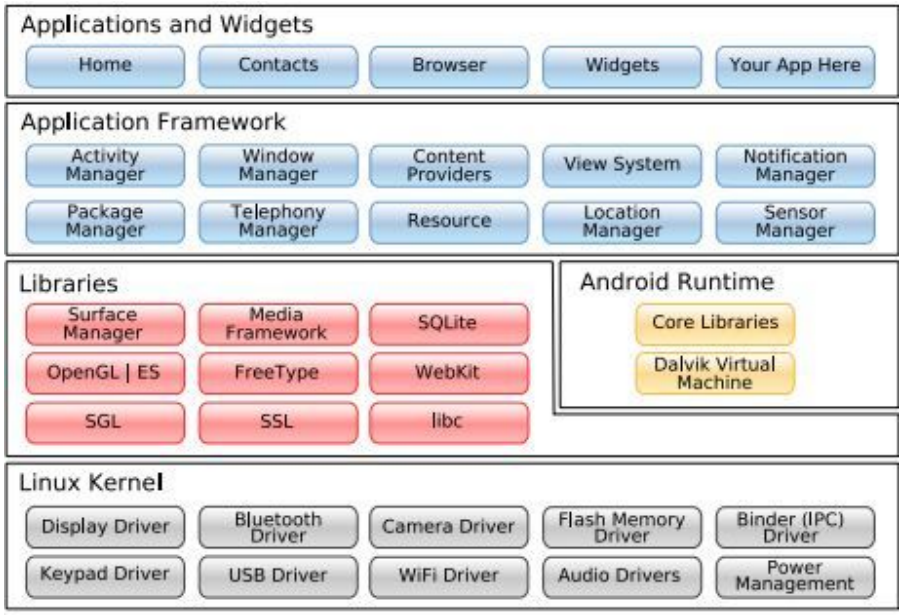
Fitur yang tersedia pada *Android* adalah:

- *Framework* aplikasi: memungkinkan penggunaan dan pemindahan dari komponen yang tersedia.
- *Dalvik virtual machine*: mesin virtual yang dioptimalkan untuk perangkat *mobile*.
- Grafik: grafik 2D dan grafik 3D yang didasarkan pada *library OpenGL*.
- *SQLite*: untuk penyimpanan data
- Mendukung media: audio, video, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- *GSM, Bluetooth, EDGE, 3G*, and *WiFi* (tergantung *hardware*).

• Kamera, *Global Positioning System* (GPS), kompas, and *accelerometer* (tergantung *hardware*).

• Lingkungan pengembangan yang kaya, termasuk *emulator*, peralatan *debugging*, dan *plugin* untuk *Eclipse IDE*.

Sistem operasi *Android* dibangun berdasarkan kernel *Linux* dan memiliki arsitektur seperti gambar di bawah ini.



Gambar 2.4 Arsitektur *Android*

Sumber: (http://www.android-indonesia.com/home/15-developers/12156-arsitektur-android)

1. *Applications*

   Serangkaian aplikasi yang ada pada perangkat *mobile* berada pada lapisan aplikasi. Aplikasi inti yang telah terdapat pada *Android* termasuk kalender, kontak, SMS, dan lain sebagainya ditulis dengan bahasa pemrograman *Java*.

2. *Applications Framework*

Pengembang aplikasi memiliki akses penuh ke *Android*sama dengan aplikasi inti yang telah tersedia. Arsitektur aplikasi ini dirancang untuk menyederhanakan penggunaan kembali komponen. Dasar dari aplikasi adalah seperangkat layanan dan sistem, yaitu berbagai *view* yang digunakan untuk membangun *user interface*, *content provider* yang memungkinkan aplikasi berbagi data, *resource manager*yang menyediakan akses bukan kode seperti grafik, *string*, dan *layout*, *notification manager*, yang akan membuat aplikasi dapat menampilkan tanda pada *status bar* dan *activity manager* yang berguna mengatur daur hidup dari aplikasi.

3. *Libraries*

   Satu set *libraries* dalam bahasa C/C++ yang digunakan oleh berbagai komponen pada sistem *Android*.

4. *Android Runtime*

   Satu set *libraries* inti yang menyediakan sebagian besar fungsi yang tersedia di *libraries* inti dari bahasa pemrograman *Java*. Setiap aplikasi akan berjalan sebagai proses sendiri pada *Dalvik Virtual Machine* (VM).

5. *Linux Kernel*

   Android bergantung pada *Linux* versi 2.6 untuk layanan sistem inti seperti keamanan, manajemen memori, manajeman proses, *network stack*, dan model *driver*.*Kernel* juga bertindak sebagai lapisan antara *hardware* dan seluruh *software*.

   (Hermawan S. 2011, p 5-7)

2.6.3 Kelebihan Android

Android banyak diminati oleh para *programmer* karena adanya *Software Development Kits* (SDK), dilengkapi dengan *emulator* yang membantu untuk menguji coba aplikasi yang dibuat serta dokumentasi yang lengkap.Selain itu, tidak ada biaya lisensi untuk memperoleh SDK ini.

Android juga telah menyediakan *Android Market* bagi para pengembang untuk menempatkan dan menjual aplikasi yang dibuatnya.Kelebihan lain yang menjadi pembeda antara *Android* dengan yang lainnya yaitu:

- Pertukaran data dan komunikasi antar proses

  Dengan adanya *intent* dan *content provider* memudahkan pengembang untuk berbagi data maupun berkomunikasi antar proses.

- Semua aplikasi sama

  Tidak ada perbedaan antara aplikasi yang dikembangkan oleh pihak ketiga dengan aplikasi inti bawaan *Android*.

- Aplikasi *services* yang berjalan di *background*

  Android memungkinkan sebuah aplikasi berjalan di *background* dan berjalan diam-diam dengan aplikasi lainnya.

- Dukungan *Google Map*

  Android telah menyediakan *Google API* yang bisa digunakan dengan mudah untuk menampilkan dan mengatur peta lokasi.(Hermawan S. 2011, 8)

2.7 Waterfall

Dalam membuat suatu program aplikasi terdapat beberapa model proses, salah satu diantaranya adalah *Waterfall Model* atau yang biasa dikenal dengan *Classic Life Cycle*. Model ini disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan (Pressman, 2010, p 39).

Terdapat 5 tahapan dalam model *Waterfall* yang akan digunakan pada aplikasi *AndroidSpeech to Text*Bahasa Indonesia dan Bahasa Inggris, yaitu:

- Komunikasi

  Diawali dengan mencari kebutuhan dari keseluruhan sistem yang akan diaplikasikan ke dalam program aplikasi.

- Perencanaan

  Proses pengumpulan elemen sistem ditingkatkan dan dipusatkan secara khusus untuk mengerti karakteristik dari program yang akan dibuat.

- Desain

  Proses perancangan menerjemahkan kebutuhan elemen sistem yang direpresentasikan ke dalam sebuah *blueprint* yang dapat diperkirakan kualitasnya sebelum dilakukan pengkodean.

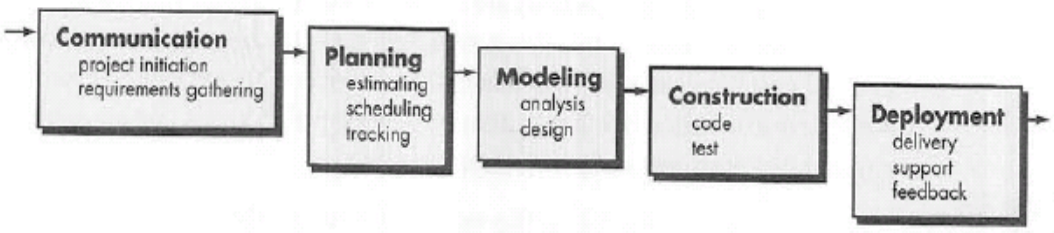
- Pengkodean

  Untuk dapat dimengerti komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh komputer, yaitu ke dalam bahasa pemograman melalui proses pengkodean. Tahap ini merupakan implementasi dari tahap desain yang secara teknis nantinya dikerjakan oleh

programmer.Setelah pengkodean maka dilakukan pengujian.Pengujian dimaksudkanuntuk memastikan hasilnya sesuai kebutuhan yang sudah didefinisikan sebelummya.

- Penyebaran

  Pemeliharaan suatu program diperlukan karena program yang dibuat tidak selamanya hanya seperti itu.Pengembangan merupakan salah satu bagian dari pemeliharaan.Pengembangan terjadi untuk memperbaiki *error* kecil yang tidak ditemukan sebelumnyaatau menerima umpan balik dari para pengguna sehingga terjadi adanya penambahan fitur-fitur yang belum ada pada program tersebut.Setelah melakukan serangkaian pengujian dan pengembangan maka program aplikasi siap untuk disebarkan ke masyarakat luas.



Gambar 2.5Model *Waterfall*
Sumber: (Pressman 2011, p39)