UNIVERSITAS INDONESIA

SISTEM *TEXT-TO-SPEECH* DENGAN METODE *UNIT SELECTION SYNTHESIS* UNTUK BAHASA INDONESIA

SKRIPSI

BAYU G. WUNDARI
04 05 03 0184

FAKULTAS TEKNIK
PROGRAM STUDI ELEKTRO
DEPOK
DESEMBER 2009

UNIVERSITAS INDONESIA

SISTEM *TEXT-TO-SPEECH* DENGAN METODE *UNIT SELECTION SYNTHESIS* UNTUK BAHASA INDONESIA

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

BAYU G. WUNDARI
04 05 03 0184

FAKULTAS TEKNIK
PROGRAM STUDI ELEKTRO
DEPOK
Desember 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : Bayu G. Wundari
NPM : 0405030184
Tanda Tangan :

Tanggal : 15 Desember 2009

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Bayu G. Wundari
NPM : 0405030184
Program Studi : Teknik Elektro
Judul Skripsi : Sistem Text-to-Speech Dengan Metode Unit Selection Synthesis Untuk Bahasa Indonesia

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Prof. Ir. Dadang Gunawan, M.Eng., Ph.D ( )

Penguji : Dr. Ir. Arman Djohan Diponegoro ( )

Penguji : Filbert Hilman Juwono, ST, MT ( )

Ditetapkan di : Depok
Tanggal : 15 Desember 2009

KATA PENGATAR

Puji syukur penulis sampaikan kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya yang begitu nyata kepada penulis sehingga skripsi ini dapat diselesaikan. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Program Studi Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

1. Bapak Prof. Ir. Dadang Gunawan, M.Eng., Ph.D Sebagai dosen pembimbing yang telah meluangkan waktunya untuk memberikan arahan, bimbingan dan diskusi sehingga skripsi ini dapat diselesaikan dengan baik dan tepat pada waktunya
2. Kedua orang tua dan keluarga saya yang telah memberikan bantuan dukungan material dan moral;
3. Teman-teman elektro angkatan 2005, F.X. Ferdinand, Marvin Yonatan.
4. Sahabat-sahabat beserta pihak lain yang baik secara langung maupun tidak langsung telah banyak membantu saya dalam menyelesaikan skripsi ini.

Akhir kata, kiranya Tuhan memberkati semua pihak atas segala kebaikan yang diberikan kepada penulis. Penulis berharap penulisan skripsi ini dapat memberikan manfaat kepada pengembangan ilmu pengetahuan.

Depok, 15 Desember 2009

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Bayu G. Wundari

NPM : 0405030184

Program Studi : Teknik Elektro

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Non-eksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Sistem *Text-to-Speech* Dengan Metode *Unit Selection Synthesis* Untuk Bahasa Indonesia

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 15 Desember 2009
Yang menyatakan

(Bayu G. Wundari)

ABSTRAK

Nama : Bayu G. Wundari
Program Studi : Teknik Elektro
Judul : Sistem Text-To-Speech Dengan Metode Unit Selection Synthesis Untuk Bahasa Indonesia

Skripsi ini membahas tentang sistem *Text-to-Speech* (TTS) untuk Bahasa Indonesia dengan *Unit Selection Synthesis* sebagai metodenya untuk mensintesa ucapan. Unit yang yang digunakan pada sistem TTS ini berupa suku kata Bahasa Indonesia. Sistem TTS yang dibuat pada skripsi ini memiliki 2 modul utama, yaitu modul *Natural Language Processing* (NLP) dan modul *Digital Signal Processing* (DSP). Modul NLP bertugas untuk memroses *input* teks yang masuk guna mendapatkan informasi dari teks itu berupa unit suku kata dengan *pitch* dan ToBI (*Tone and Break Indices*) yang bersesuaian dengan kalimat pada teks masukan, Informasi ini kemudian digunakan oleh modul DSP untuk menghasilkan ucapan. Pada modul DSP ini, metode sintesa ucapan yang digunakan adalah *Unit Selection Synthesis* yang merupakan generasi ketiga setelah *Concatenative Synthesis*. Metode *Unit Selection Synthesis* menggunakan *database* yang sangat banyak sekali untuk dapat menghasilkan ucapan dengan tingkat kealamian yang tinggi. Untuk tiap unit suku kata memiliki karakteristik seperti *pitch*, durasi, *Mel Frequency Cepstrum Coefficient* (MFCC), dan ToBI yang berbeda-beda dengan unit yang lain walaupun suku kata yang digunakan adalah sama. Suku kata dengan karakteristik yang berbeda tersebut diperoleh dari hasil pemotongan *file wav* suatu rekaman ucapan.

Dari segi intellijibilitas, ucapan yang dihasilkan tidaklah baik. Hal ini disebabkan *database* yang dimiliki sangat kurang dan rekaman ucapan yang dijadikan sumber data memiliki banyak noise sehingga mengganggu proses pemotongan *file wav* untuk mendapatkan suku kata. Namun tingkat kealamian ucapan yang diperoleh dari sistem TTS ini dapat dikatakan cukup baik karena *pitch* dari suku kata yang cukup bervariasi sehingga intonasi yang terdengar tidak mendatar saja.

Kata kunci: *Text-to-Speech, Unit Selection Synthesis,* ToBI, MFCC.

ABSTRACT

Name : Bayu G. Wundari
Study Program : Electrical Engineering
Title : Text-to-Speech System with Unit Selection Synthesis Method for Bahasa Indonesia

This undergraduate thesis discusses about a Text-to-Speech system with Unit Selection Synthesis as it's method to synthesize speech. Units which are used as the units for the synthesizer are Bahasa Indonesia syllables. In this study, the TTS system uses 2 main modules, they are Natural Language Processing module (NLP) and Digital Signal Processing Module (DSP). The NLP module processes input text for retrieving information from the input in the form of syllables with their pitch and ToBI (Tone and Break Indices) associated with the sentences in the text. The retrieved information then used by DSP module to produce speech. The third generation synthesizer after concatenative synthesis, Unit Selection Synthesis, is chosen as the speech synthesizer in the DSP module. To get speech with high naturalness, the synthesizer must uses a large speech database. Each and every syllable has it's own characteristics such as pitch, duration, Mel Frequency Cepstrum Coefficient (MFCC), and ToBI that are different from other units eventhough the syllables are the same. The author get the syllables by trimming a wav file of recorded speech.

From the intelligibility point of view, the quality of the produced speech is not good. It is because the quality of the possessed database is poor and the recorded speech chockablock with noise in such a way that unsettles the process of trimming the wav file in order to get the syllables. Yet, from the naturalness point of view, the quality of the speech could be accepted because of the variety of the pitch of the syllables so that the perceived speech is not monotone.

Keywords: Text-to-Speech, Unit Selection Synthesis, ToBI, MFCC.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Speech Synthesis adalah suatu studi mengenai bagaimana membuat suatu mesin, khususnya komputer, untuk dapat membaca dengan mengeluarkan suara. Dengan demikian, proses tersebut terkait dengan tiga hal, yaitu proses membaca, proses berbicara, dan perihal yang terkait untuk membuat mesin tersebut dapat melakukan kedua hal itu. Sedang *Text-to-Speech* merupakan suatu proses mengkonversi teks tertulis (*written text*) menjadi ucapan (*speech*). Dalam komunitas enjinering, TTS sering disebut juga sebagai *speech synthesis*.

Manusia telah lama termotivasi untuk menciptakan mesin yang dapat berbicara. Usaha pertama untuk menciptakan mesin tersebut termanifestasi pertama kali dalam bentuk model mekanik yang menirukan alat-alat ucap manusia. Dua contohnya dapat ditelusuri pada abad ke-13 ketika filsuf Jerman, Albertus Magnus dan ilmuwan Inggris, Roger Bacon membuat prestasi dengan menciptakan kepala besi yang dapat berbicara (*metal talking heads*). Sayangnya, tidak ada dokumen yang memadai mengenai divais ini. Lima ratus tahun kemudian, barulah terdapat dokumen mengenai usaha manusia dalam menciptakan mesin yang dapat berbicara, tepatnya pada tahun 1759. Di tahun tersebut, Ch. G. Kratzenstein, seorang professor fisiologi di Copenhagen, menciptakan rongga resonan (*resonant cavities*) yang dieksitasi dengan suatu vibrator sehingga dapat menghasilkan suara 5 vowel (huruf hidup) a, i, u, e, dan o. Dan sekitar pada tahun yang sama, ditemukannya mesin penghasil ucapan manusia di oleh Wolfgang von Kempelen. Mesin buatan von Kompelen ini mampu menghasilkan ucapan sebuah kata secara penuh dan mampu mengucapkan sebuah kalimat pendek. Kemudian pada tahun 1835, rekonstruksi dari mesin ini dilakukan oleh Wheatstone di Dublin. Mesin hasil rekonstruksi ini berbeda dengan buatan von Kempelen dalam hal rongga mulut dan pengontrolan suara. Namun kekurangan dari mesin ini adalah kurangnya mekanisme pengontrolan *pitch*. Dilanjutkan oleh Joseph Faber, rekonstruksi yang ia kerjakan dapat dikatakan mengalami cukup kemajuan. Mesin hasil rekonstruksinya memiliki

model dari lidah dan rongga *pharyngeal* yang bentuknya dapat dikontrol.

Dilanjutkan oleh R. R. Riesz (USA) pada tahun 1937, ia membuat divais yang mirip seperti yang telah dijelaskan di atas, namun dengan bentuk *vocal tract* yang bentuknya hampir alami. Dengan berkembangnya teknologi pada awal abad ke-20, mesin penghasil suara ini dapat dikembangkan sehingga dapat dilakukan pensintesaan pada suara ucapan yang dihasilkan dengan menggunakan alat elektronika. Divais pertama yang dipublikasikan dikenal sebagai VODER yang dikembangkan oleh Homer Dudley. Divais ini dipresentasikan pada *World Fair* tahun 1939 di New York.

Akhirnya, sejak tahun 1970, perkembangan lebih mendalam lagi dalam *speech synthesis* dikaitkan dengan teknologi komputer. Sekarang ini, tidak cukup hanya mensimulasikan *speech* yang alami dengan rangkaian listrik, namun rangkaian listrik itu juga ikut disimulasikan. Komputer memungkinkan *speech synthesis* untuk beberapa kegunaan praksis, dan beberapa sistem dengan kemampuan mengkonversi teks menjadi *speech* dikembangkan. Dan sistem ini kemudian dikenal sebagai *Text-to-Seech System*.

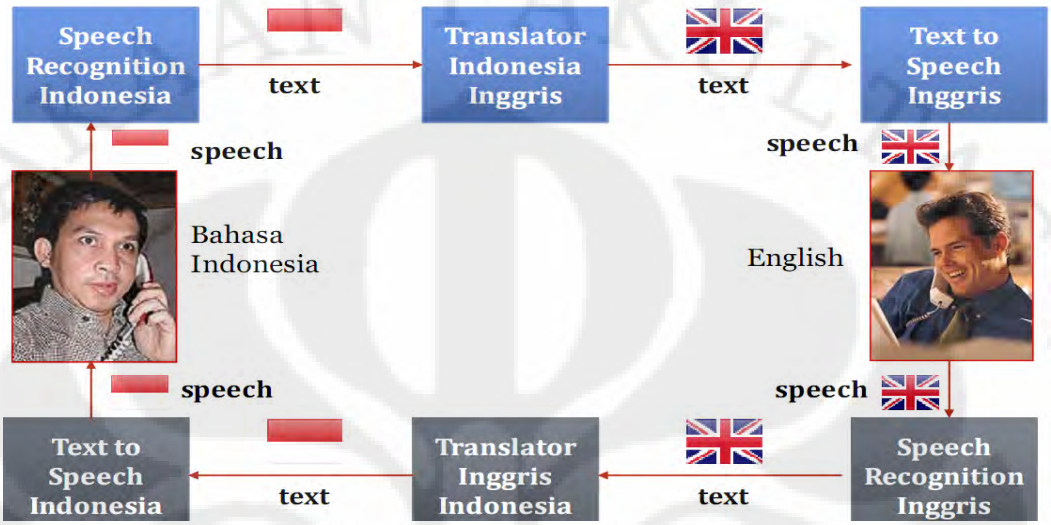
Sistem *Text-to-Speech* (TTS *system*) memiliki banyak kegunaan. Penggunaan utama dari sistem ini memang dikhususkan bagi tunanetra sebagai suatu sistem untuk membaca teks dari sebuah buku kemudian mengkonversinya menjadi *speech*. Namun, seiring berjalannya waktu, sistem ini berkembang hingga munculnya sistem TTS yang terintegrasi dengan sistem komputer sehingga para tunanetra dapat diarahkan untuk menggunakan komputer selayaknya orang awas (orang yang dapat melihat). Tapi TTS tidak sebatas kegunaannya pada bidang itu saja, seperti:

1. ***Speech-to-Speech Translation***

   Aplikasi ini berfungsi untuk mengkomunikasikan seseorang dengan satu bahasa tertentu dengan orang lain yang berbahasa asing. Misalnya, ketika orang Indonesia sedang menelepon dengan orang Inggris. Pada aplikasi ini, sistem TTS berfungsi untuk menghasilkan *speech* yang berbahasa sesuai dengan negara masing-masing. Sistem TTS mengambil *input* dari sistem translator bahasa yang mana keluaran dari sistem ini berupa teks. Kemudian,

PERPUSTAKAAN FAKULTAS TEKNIK
UNIVERSITAS INDONESIA

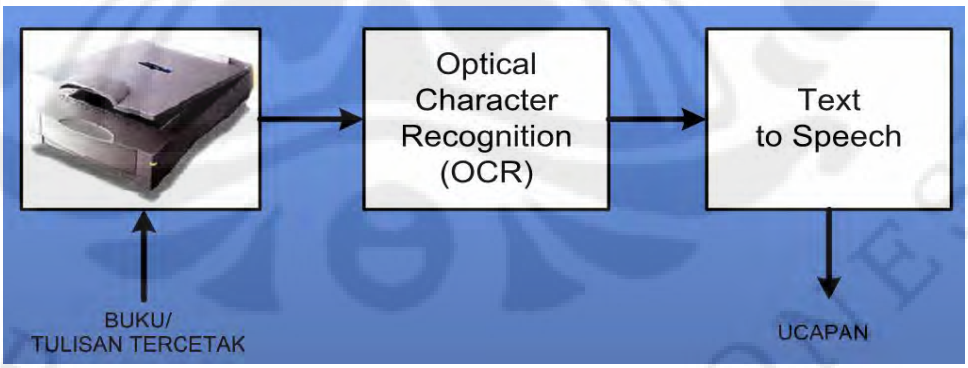
teks ini akan diproses pleh sistem TTS untuk menghasilkan *speech*. Untuk lebih jelasnya, dapat dilihat pada Gambar 1.1 berikut:



Gambar 1.1. Speech-to-Speech Translation [1].

2. ***Document Reader***

   Document reader adalah alat untuk membacakan dokumen. Aplikasi ini berguna sekali untuk membantu khususnya mereka yang tunanetra dalam membaca dokumen-dokumen. Sistem TTS pada aplikasi ini mengambil *input* dari sistem OCR (*Optical Character Recognition*) lalu *input* tersebut akan diproses untuk menghasilkan *speech*.

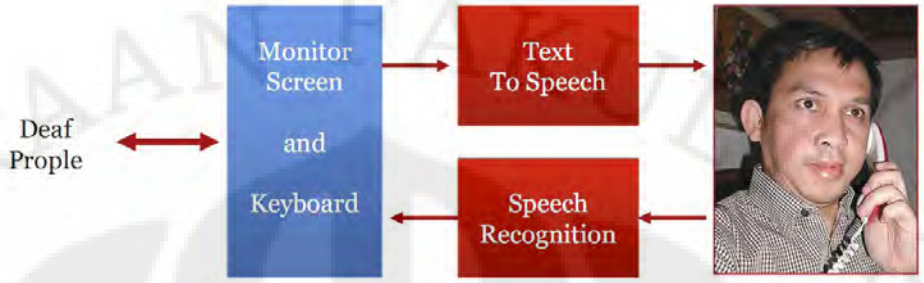


Gambar 1.2. Document Reader [1].

3. ***Terminal for Deaf People***

   Sesuai dengan namanya, aplikasi ini berfungsi sebagai alat komunikasi bagi orang tuli sehingga dapat berinteraksi dengan orang yang tidak tuli ketika

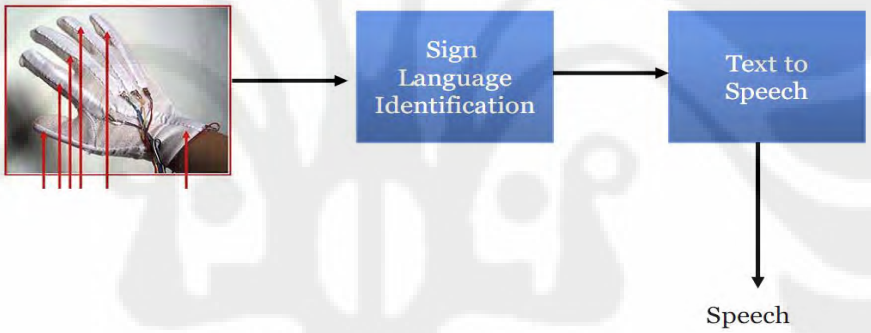
bertelepon. Sistem TTS perlu dihubungkan dengan komputer dan telepon sebagai perantara.



Gambar 1.3. Terminal fo Deaf People [1].

4. ***Talking Aid***

Talking aid adalah alat untuk menghasilkan *speech* dari *input* yang berupa bahasa isyarat (*sign language*).



Gambar 1.4. Talking aid [1].

5. ***Screen Reader***

Aplikasi ini sudah diterapkan pada komputer-komputer sekarang ini. Bagi tunanetra, aplikasi jenis ini sangat berguna sekali dalam membantu mereka ketika berinteraksi dengan komputer.

6. ***Interaksi secara lisan dengan komputer***

Seperti pada film "Star Trek", aplikasi ini memungkinkan manusia untuk berinteraksi dengan komputer secara lisan. Dengan kata lain, perintah yang dimasukkan ke komputer tidak perlu lagi berasal dari keyboard atau mouse, melainkan dengan hanya berbicara saja, maka komputer dapat melakukan instruksi yang telah diperintahkan.

1.2. Tujuan Penulisan

Mungkin seseorang bertanya, ”Apakah diperlukan bagi sebuah mesin untuk membaca sambil bersuara dengan kualitas suara yang tinggi dan dapat berbicara seperti manusia? Bukankah cukup bila mesin dapat membaca sambil bersuara saja, tidak perlu seperti manusia?” Pengalaman menunjukkan bahwa orang sangat sensitif, tidak hanya dengan kata-kata yang diucapkan, tapi juga cara mesin itu berbicara. Kebanyakan orang merasa bahwa suara mesin berkualitas rendah tidak nyaman untuk didengar dan menyakitkan. Dan pengalaman juga menunjukkan bahwa pengguna menginginkan sistem TTS dengan suara yang mirip dengan manusia. Berangkat dari permasalahan tersebut, dibuatlah skripsi ini, yaitu bertujuan untuk memberikan pembahasan bagaimana membangun suatu sistem yang dapat membaca teks berbahasa Indonesia dan mengeluarkan *speech* yang mirip dengan manusia. Kualitas sistem ini diukur dari segi intellijibilitas dan tingkat kealamian (*naturalness*).

1.3. Pembatasan Masalah

Permasalahan bagaimana menghasilkan suara yang mirip manusia masih memiliki maksud yang luas. Misalnya mengenai emosi yang terdapat di dalam suatu teks, apakah di dalam teks tersebut mengandung unsur kesedihan, kemarahan , kebahagiaan, dan sebagainya. Dengan demikian, pada skripsi ini, permasalahan tersebut penulis batasi hanya sampai bagaimana suatu teks dibaca oleh sistem sehingga pembacaan itu memiliki melodi yang cukup, tidak seperti robot. Dan karena keterbatasan dalam membuat *database*, kalimat yang dapat diucapkan oleh sistem yang dibuat baru dapat mengucapkan kalimat tertentu saja. Unsur emosi yang ada dalam suatu teks tidak penulis sertakan di dalam skripsi ini. Selain itu, pembatasan juga dilakukan pada masalah semiotik yang terkandung di dalam suatu teks. Masalah semiotik pada teks yang dapat dibaca oleh sistem dibatasi pada tulisan berupa penanggalan, waktu, dan angka.

1.4. Metode Penelitian

Adapun metode penulisan yang penulis lakukan dalam menyusun skripsi ini adalah:

1. Dengan melakukan studi literature dan kepustakaan dari situs-situs internet berupa buku, *e-book*, jurnal, *mailing list*, dan artikel-artikel yang terkait dengan system TTS.
2. Mendisain dan memrogram sistem TTS.
3. Diskusi dengan dosen pembimbing tugas akhir dan teman-teman.

1.5. Sistematika Penulisan

Skripsi ini terdiri dari enam bab. Bab I berisi pendahuluan yang menjelaskan: latar belakang masalah, tujuan penulisan, pembatasan masalah, metodologi penulisan, dan sistematika penulisan. Bab II membahas mengenai apa itu suara, bagaimana suara ucapan (speech) dapat dihasilkan, dan bagaimana proses komunikasi dapat berlangsung. Bab III membahas apa itu system TTS beserta dengan beberapa metode dalam mensintesis ucapan. Bab IV berisi tentang perancangan system dan algoritma dari program TTS yang dibuat. Bab V membahas mengenai pengujian dan analisis dari system TTS yang sibuat. Bab VI berisi kesimpulan dari pembahasan tugas akhir.

BAB II

Properti Suara Pada Manusia

2.1. Suara

Suara merupakan suatu bagian yang sangat penting bagi banyak makhluk hidup. Hewan dapat menggunakan suara untuk memburu mangsanya, menarik perhatian dari pasangan, dan memberi peringatan akan adanya pemangsa [2]. Bagi manusia, suara sirine dapat meningkatkan kewaspadaan kita akan adanya bahaya, sedang suara musik dapat memberikan kita kelegaan. Dari pengalaman kita sehari-hari, kita telah mengenal karakteristik dari suara, seperti volume, nada, dan *pitch* (berkaitan dengan frekuensi) suara. Tanpa memikirkan semuanya ini dan karakteristik lainnya, kita telah seringkali mengelompokkan suara-suara yang kita dengar, misalnya suara ketika kita sedang berbicara dan suara music [2].

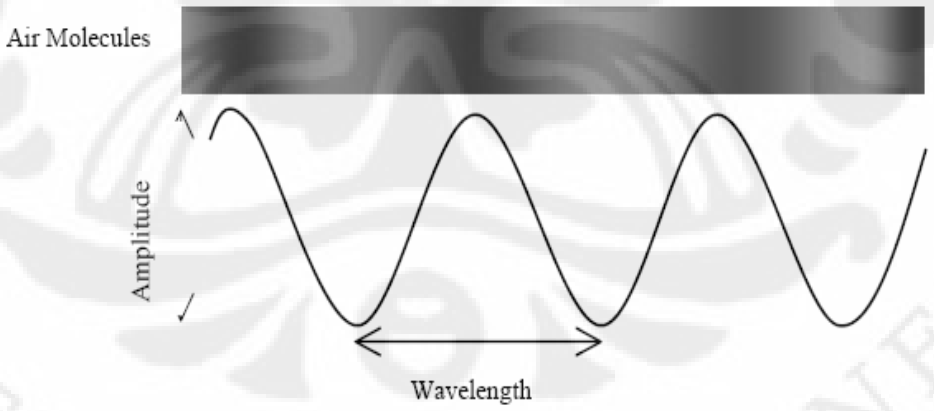
Suara adalah suatu gelombang longitudinal yang dihasilkan dari peregangan dan perapatan molekul udara [2]. Karena suara adalah gelombang longitudinal, maka arah rambatnya adalah sejajar dengan energi yang dikenakan pada molekul udara tersebut. Daerah perapatan (*compressions*) adalah zona di mana molekul udara mengalami suatu gaya akibat dari energi tersebut sehingga molekul udara itu membentuk konfigurasi yang lebih rapat daripada biasanya. Sedang peregangan (*rarefactions*) adalah suatu zona di mana molekul udara membentuk konfigurasi yang tidak rapat.

Proses terbentuknya suara yang berasal dari suatu sumber suara, seperti dari pita suara, *loudspeaker*, atau sumber suara apa saja, dapat dianalogikan dengan proses seperti pada Gambar 2.1 di bawah ini.

Gambar 2.1. Analogi proses pembentukan suara. (a) Kondisi udara di sekitar bel ketika bel belum bergetar. (b) Kondisi udara di sekitar bel ketika bel bergetar [2].

Ketika bel bergerak maju dan mundur, bel menyebabkan udara di sekitarnya menjadi bergetar. Ketika bel bergerak maju, udara menjadi bergerak lebih cepat, sedang ketika bel bergerak ke belakang, udara menjadi bergerak lebih lambat. Hasil dari pergerakan bel yang maju menyebabkan tekanan udara menjadi lebih tinggi sedikit dari normalnya. Pergerakan bel ke belakang menyebabkan tekanan udara menjadi sedikit lebih rendah dari normalnya. Tumbukan yang terjadi antar partikel udara menyebabkan perbedaan tekanan yang dihasilkan tadi bergerak menjauhi bel ke segala arah (untuk menyederhanakan, pergerakan tekanan udara digambarkan dalam satu arah saja, padahal pergerakan tersebut mencakup ke segala arah).

Konfigurasi molekul udara yang merapat dan meregang tersebut sering digambarkan sebagai gelombang sinusoidal seperti pada Gambar 2.2 di bawah ini. Dalam hal ini, puncak dari gelombang sinusoidal menggambarkan molekul udara yang merapat, dan lembah gelombang sinusoidal menggambarkan molekul udara yang meregang.



Gambar 2.2. Perubahan konfigurasi gelombang suara yang dapat direpresentasikan sebagai gelombang sinusoidal [3].

Besarnya usaha (*work*) yang dikerjakan oleh energi itu untuk membuat molekul udara bergerak dapat diestimasi dari besarnya perpindahan molekul tersebut dari posisi diamnya (*resting positions*). Dan selanjutnya tingkat

perpindahan ini dapat diestimasi dari besarnya amplitudo gelombang suara tersebut seperti pada Gambar 2.2. Pada umumnya untuk mengukur amplitudo gelombang suara itu dinyatakan dalam skala logaritmik (decibels, dB). Skala decibel merupakan perbandingan dari daya dua gelombang suara dan dapat dinyatakan sebagai berikut [3].

$$\mathbf{10\log_{10}(P1/P2)} \quad \textbf{.....(2.1)}$$

Tingkat Tekanan Suara (*Sound Pressure Level*, SPL) adalah suatu ukuran dari tekanan *absolute* suara P dalam decibel [3]:

$$\mathbf{SPL(dB) = 20\log_{10}}\left(\frac{P}{P_0}\right) \quad \textbf{.....(2.2)}$$

Dengan titik 0 (dB) berkorespodensi dengan batas bawah pendengaran, dimana P_0 = 0.0002 μbar untuk nada berfrekuensi 1 kHz. Percakapan pada jarak sekitar 3 kaki adalah sekitar 60 dB SPL. Selain dalam skala decibel, satuan Watts/meter^2 juga sering digunakan sebagai satuan intensitas suara [3].

Batas bawah pendengaran seseorang adalah 10^{-12} W/m^2. Batas bawah pendengaran ini disebut sebagai *threshold of hearing (TOH)*. Besarnya intensitas ini seperti sebuah gelombang yang menyebabkan pergerakan molekul dalam kisaran 10^{-12} cm. Suara terkeras yang dapat didengar oleh manusia tanpa mengalami kerusakan fisik adalah 10^{12} kali dari TOH. Suara terpelan yang dapat didengar adalah 0 dB, dan suara terkeras yang masih dapat ditoleransi untuk didengar oleh manusia adalah 120 dB. Tabel 2.1 memperlihatkan daftar intensitas dan tingkat decibel dari beberapa jenis suara:

Tabel 2.1. Daftar intensitas dan tingkat decibel dari beragam jenis suara [3].

Sound	dB Level	Times > TOH
Threshold of hearing (TOH: $10^{-12} W/m^2$)	0	10^0
Light whisper	10	10^1
Quiet living room	20	10^2
Quiet conversation	40	10^4
Average office	50	10^5
Normal conversation	60	10^6
Busy city street	70	10^7
Acoustic guitar – 1 ft. away	80	10^8
Heavy truck traffic	90	10^9
Subway from platform	100	10^{10}
Power tools	110	10^{11}
Pain threshold of ear	120	10^{12}
Airport runway	130	10^{13}
Sonic boom	140	10^{14}
Permanent damage to hearing	150	10^{15}
Jet engine, close up	160	10^{16}
Rocket engine	180	10^{18}
Twelve feet. from artillery cannon muzzle ($10^{10} W/m^2$)	220	10^{22}

2.2. Sistem Suara Manusia

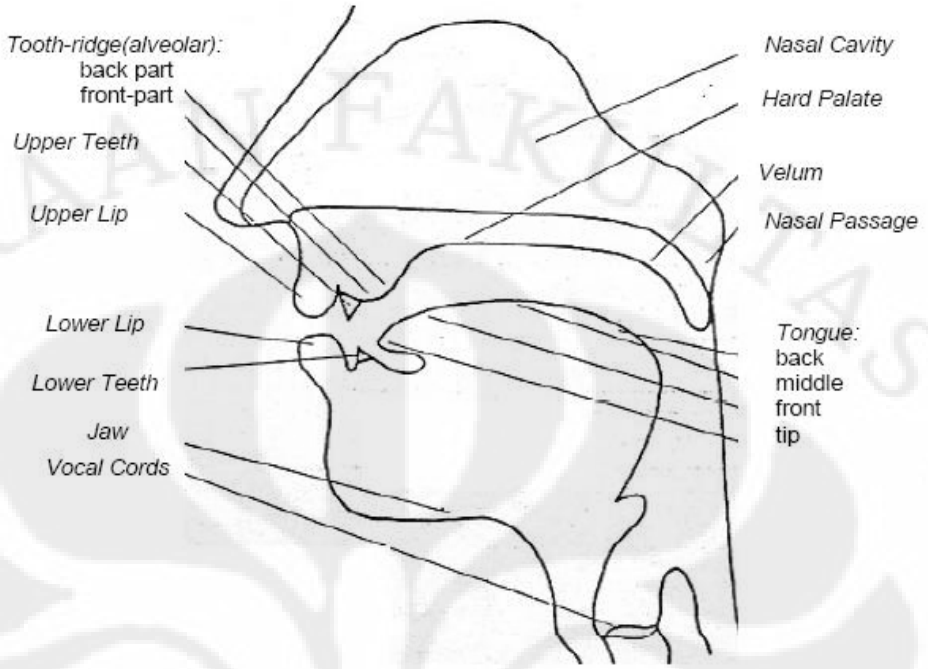
Suara manusia dihasilkan oleh gelombang udara bertekanan yang memancar dari mulut dan *nostrils* (bagian yang berada di dalam hidung) pembicara. Dalam kebanyakan dunia bahasa, khususnya bahasa Indonesia, fonem **(**fonem : satuan unit terkecil yang mampu menunjukkan kontras makna. Misalnya /h/ adalah fonem, karena membedakan makna kata *harus* dan *arus*. /b/ dan /p/ adalah dua fonem yang berbeda karena *bara* dan *para* berbeda maknanya.**)** dapat dibagi menjadi dua kelas [3] :

1. Konsonan : bunyi bahasa yang dihasilkan dengan menghambat aliran udara pada salah satu tempat di saluran suara di atas glottis (selain a, i, u, e, o).
2. Vowel (huruf hidup) : bunyi bahasa yang dihasilkan oleh arus udara dari paru-paru melalui pita suara dan penyempitan pada saluran udara di atas glottis (a, i, u, e, o).

Komponen penghasil suara yang utama dari organ penghasil suara pada manusia adalah paru-paru, trakea, laring, *pharyngeal cavity* (kerongkongan), rongga mulut dan hidung. Seperti yang digambarkan pada Gambar 2.3, organ penghasil suara pada manusia terdiri dari:

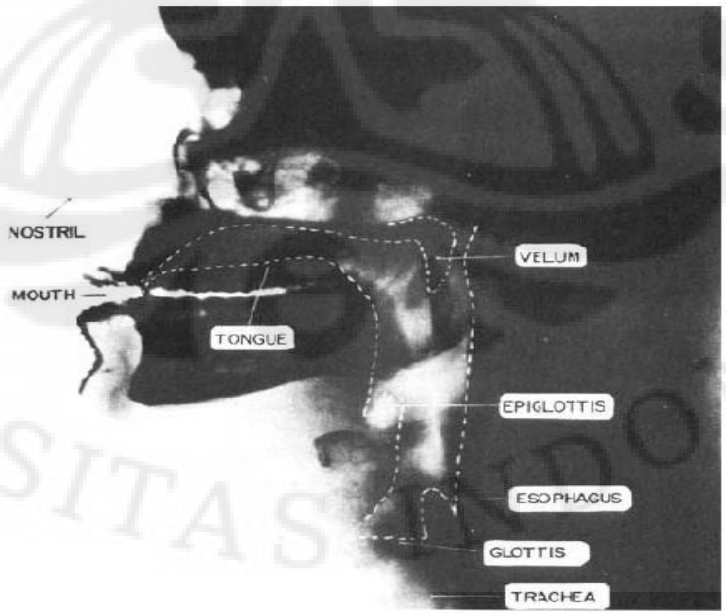
1. Paru-paru : sumber udara ketika bersuara (berbicara).
2. *Vocal Cords* (laring) : bagian atas tenggorokan yang berisi pita suara.
   Pada saat *vocal cord* berada dalam keadaan tegang, aliran udara akan menyebabkan vibrasi pada *vocal cord* dan menghasilkan bunyi ucapan yang disebut sebagai *voiced speech*.

   Pada saat *vocal cord* berada dalam keadaan lemas, aliran udara akan melalui daerah yang sempit pada vocal tract dan menyebabkan terjadinya turbulensi dan menghasilkan suara yang disebut sebagai *unvoiced speech*.
3. *Velum* (*soft palate*) : beroperasi seperti sebuah kran (*valve*), yang bila berada dalam keadaan membuka, akan menyebabkan udara masuk melalui *nasal cavity*.
4. *Hard Palete* : Bagian yang agaka panjang dank eras yang terletak di atas rongga mulut.
5. Lidah (*tongue*) : bagian artikulator yang dapat bergerak secara fleksibel.
6. Gigi (*teeth*) : Dapat berfungsi untuk membantu lidah dalam membentuk konsonan.
7. Bibir : bagian dari mulut yang dapat mempengaruhi kualitas ucapan, dan menghentikan secara penuh aliran udara untuk konsonan tertentu.



Gambar 2.3. Skematik organ penghasil suara pada manusia [3].

Paryngeal dan rongga mulut sering disebut sebagai *vocal track*. Sedang rongga hidung sering disebut sebagai *nasal track. Vocal track* dapat dilihat pada Gambar 2.4 di bawah ini:

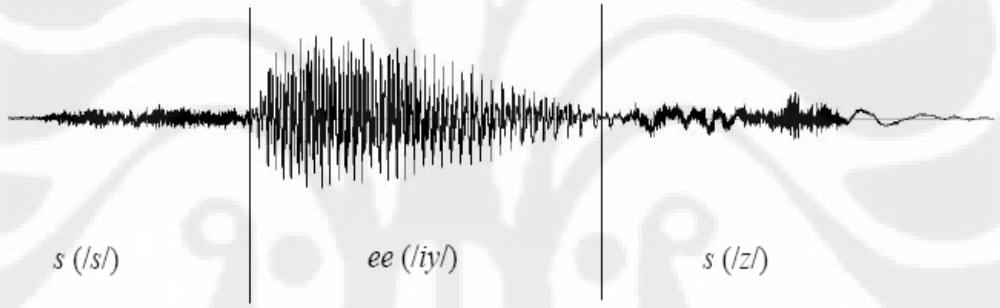


Gambar 2.4. Foto sinar X penampang alat-alat penghasil suara manusia [4].

Vocal track ditandai oleh garis putus-putus, dimulai dari *glottis* sampai dengan mulut. Panjang *vocal track* pada pria pada umumnya sepanjang 17 cm. Nasal track dimulai dari bagian belakang langit-langit dan berakhir pada *nostrils* (hidung).

2.3. Mekanisme Suara Manusia

Perbedaan yang paling mendasar antara tipe suara dalam percakapan adalah *voiced* dan *unvoiced*. Suara *voiced*, umumnya memiliki lebih banyak energi dibandingkan dengan suara *unvoiced* seperti ditunjukkan pada Gambar 2.5. Gambar 2.5 merupakan gelombang yang dibentuk dari kata *sees*, yang terdiri dari 3 buah fonem: sebuah konsonan *unvoiced* /s/, sebuah vowel /iy/, dan sebuah konsonan *voiced* /z/.

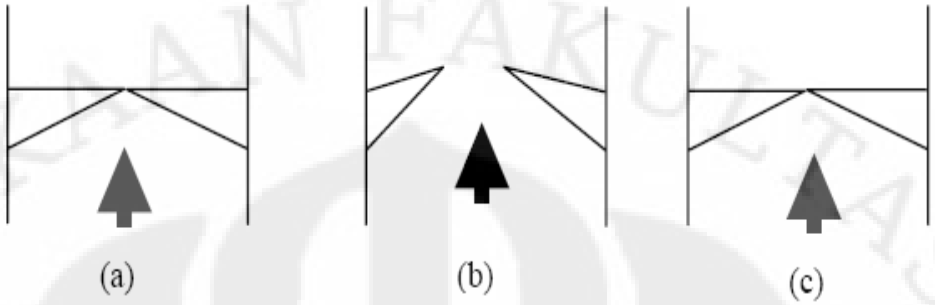


Gambar 2.5. Gelombang suara dari kata *sees* [3].

Apakah yang membuat perbedaan ini dalam mekanisme pembentukan suara pada manusia? Ketika *vocal cord* bervibrasi selama pembentukan fonem, fonem tersebut adalah *voiced*. Bila *vocal cord* tidak bervibrasi, maka fonem tersebut dikatakan *unvoiced*. Misalnya, pada kata *‚þobo'* dan *'popo'*. Bila tangan diletakkan pada tenggorokan, akan dirasakan getaran yang lebih di kerongkongan ketika diucapkan kata *‚þobo'* dibandingkan ketika diucapkan kata *‚popo*'. Kata-kata yang memiliki getaran yang lebih ketika diucapkan disebut sebagai *voiced speech*. Sedang yang satunya lagi disebut sebagai *unvoiced speech*.

Perbedaan antara karakter vowel dibentuk dari pemakaian bibir dan lidah untuk membentuk rongga resonansi mulut dalam bentuk yang berbeda-beda. *Vocal cord* dapat bervibrasi pada frekuensi yang rendah (sekitar 60 Hz) dan sampai frekensi yang lebih tinggi (sekitar 300 Hz). Frekuensi membuka dan

menutupnya *vocal cord* dalam laring disebut sebagai frekuensi dasar (*fundamental frequency*).

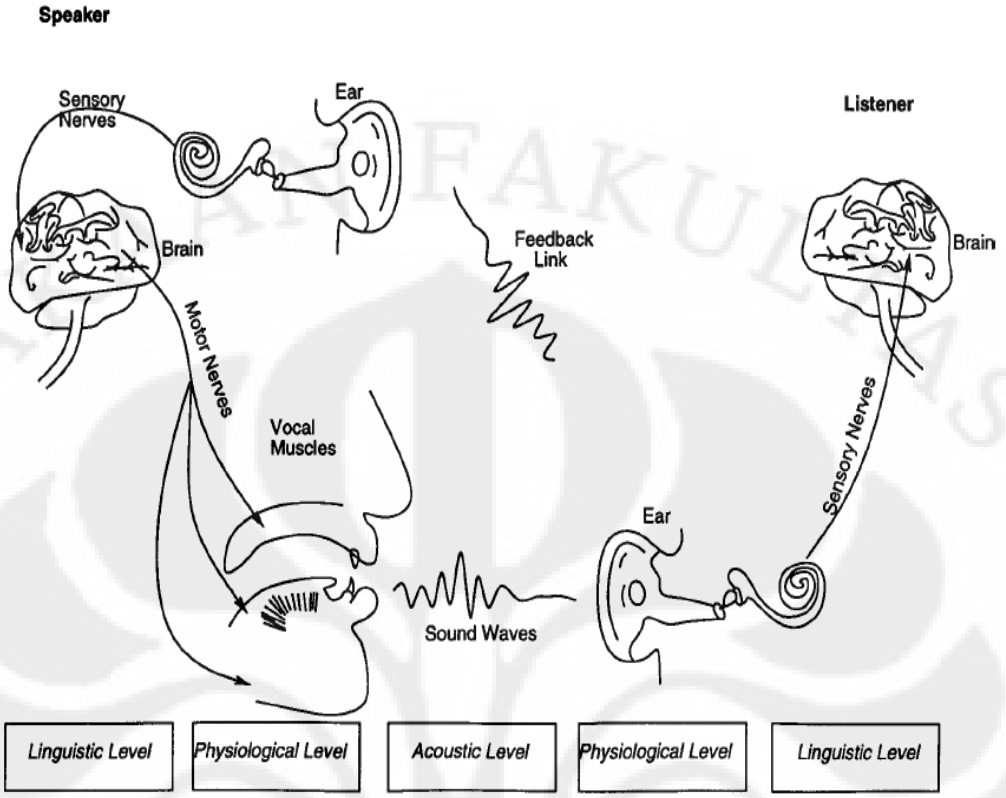


Gambar 2.6. *Vocal cord* bervibrasi pada laring. (a) *Vocal cord* menutup, sedang tekanan sub-glottal meningkat. (b) Perbedaan tekanan trans-glottal menyebabkan *vocal cord* terbuka ke atas. (c) Tekanan antara sub-glottal (bawah *vocal cord*) dan elastisitas dari *vocal cord* berada dalam keadaan seimbang, siap untuk siklus berikutnya [3]

Siklus glottal ditunjukkan pada Gambar 2.6 di atas. Pada tahap (a), *vocal cord* menutup dan aliran udara dari paru-paru digambarkan oleh tanda anak panah ke atas. Pada titik yang sama, tekanan udara di bawah *vocal cord* (*sub-glottal*) meningkat hingga melebihi tahanan dari *vocal cord* sehingga tekanan udara dari paru-paru menekan ke atas *vocal cord* sehingga membuka ke atas (b). Elastisitas dari *vocal cord* cenderung untuk mengembalikan *vocal cord* ke keadaan semula ketika tekanan udara dan elastisitas dari *vocal cord* berada dalam keadaan sama kuat (c).

2.4. Proses Komunikasi

Sebuah cara yang dapat membantu untuk mendemonstrasikan apa yang terjadi selama percakapan tengah berlangsung adalah dengan menggambarkan sebuah contoh sederhana mengenai dua orang berbicara satu sama lainnya. Salah satu dari mereka, mentransmisikan informasi kepada yang lain, yaitu si pendengar. Rantai peristiwa ini disebut sebagai *‚rantai percakapan'* (*the speech chain*), dan digambarkan pada Gambar 2.7 di bawah ini:



Gambar 2.7. *The Speech Chain* [5].

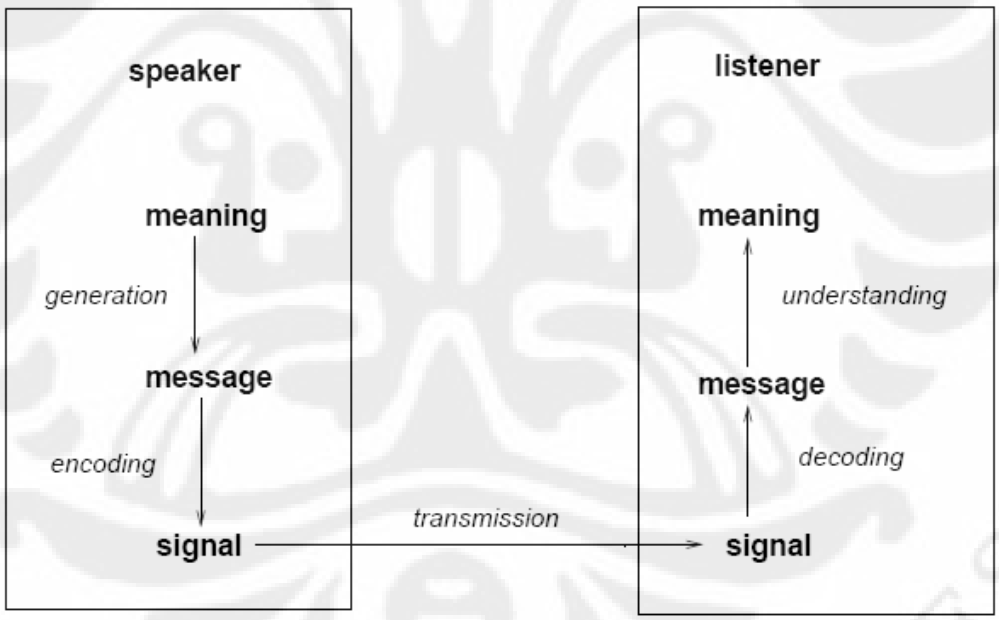
Si pembicara *(speaker*) pertama kali menentukan terlebih dahulu akan hal apa yang ingin diucapkan. Lalu hal yang ingin diucapkan tersebut dituangkan dalam bentuk linguistik (*linguistic form*), dengan menyusun kata-kata atau klausa-klausa yang disusun sesuai dengan aturan bahasa (*grammar*) yang tepat. Proses ini berkaitan dengan otak si pembicara di mana pada tempat ini, instruksi-instruksi yang bersesuaian akan dikirim ke otot yang mengendalikan organ suara (*vocal organ*, seperti: lidah, bibir, dan pita suara) melalui syaraf motor (*motor nerve*). Impuls dari syaraf ini akan menyebabkan *vocal organ* bergerak sedemikian rupa sehingga menghasilkan perubahan kecil tekanan udara di sekitarnya dan perubahan tekanan udara tersebut akan dipropagasikan melalui udara dalam bentuk gelombang suara (*sound wave*).

Gelombang suara tersebut sampai ke telinga pendengar dan mengaktifkan mekanisme pendengaran di pendengar tadi. Mekanisme pendengaran di dalam telinga pendengar menghasilkan impuls syaraf yang akan menjalar melalui syaraf akustik (*acoustic nerve* atau *sensory nerve*) sampai ke otak pendengar. Ketika

impuls syaraf sampai ke otak melalui syaraf akustik, aktivitas otak yang terkait dengan pendengaran akan dikuatkan oleh impuls syaraf di telinga. Perubahan aktifitas yang terjadi pada otak menyebabkan terjadinya pengenalan dan pemahaman terhadap pesan yang disampaikan oleh pembicara.

Syaraf pendengaran dari si pembicara tadi juga menyediakan *feedback* (umpan balik) ke telinga pembicara itu sendiri. Otak secara berkesinambungan membandingkan kualitas suara yang dihasilkan dengan kualitas suara yang diinginkan, sehingga dapat dibuat suatu penyesuaian yang diperlukan agar sesuai dengan suara yang diinginkan. Kecacatan dalam penerimaan *feedback* ini menjadi salah satu alasan mengapa orang yang mengalami kesulitan dalam pendengaran mengalami kesulitan dalam berbicara yang benar dan jelas.

Jadi, proses komunikasi antara dua orang (dialog) dapat dijelaskan pada Gambar 2.8 berikut:



Gambar 2.8. Proses yang terjadi di dalam percakapan antara 2 orang [6].

Misalkan si pembicara adalah si A, dan si pendengar adalah B. Anggaplah A memutuskan untuk mengatakan suatu hal kepada B. Untuk melakukan hal ini, A harus menghasilkan terlebih dahulu kata-kata atau frasa-frasa yang akan disampaikan (*linguistic level*). Kata-kata atau frasa-frasa tersebut disebut sebagai pesan (*message*). Kemudian, A meng-*encode message* itu menjadi *signal* ucapan

(*speech signal*) yang nantinya akan dikirim ke B. Proses *encoding* ini berada pada tahap fisiologi (*Physiological level*). Kemudian *speech signal* itu diterima oleh B dan oleh B akan di-*decode* menjadi bentuk *message*. Proses pengiriman *speech signal* ini berada pada tahap akustic (*acoustic level*). Sedang proses *decoding* pada B berada pada tahap fisiologi (*physiological level*). Kemudian B akan membentuk kata-kata atau frasa-frasa yang dapat dimengerti olehnya untuk dapat mengerti isi dari *message* itu. Dan proses ini berada pada tahap linguistik (*linguistic level*).

Dengan demikian, ucapan bermula dari tahap linguistik (*linguistic level*) dari *speech chain* di dalam otak si pembicara melalui pemilihan kata-kata dan frasa-frasa dan berakhir pada *linguistic level* dari si pendengar di dalam otaknya yang mengkonversi aktifitas syaraf yang dibawa melalui syaraf akustik menjadi bahasa yang dapat dimengerti oleh si pendengar. Dari *linguistic level*, kemudian ke *physiological level* (tahap fisiologi, tahap dimana ucapan itu dilafalkan atau diterima) ketika kata-kata dan frase-frase tersebut diucapkan. Setelah itu ke *acoustic level* (tahap pendengaran). Pendengar kemudian membawanya ke *physiology level* selama proses pendengaran dan dilanjutkan ke linguistik level dimana sensasi yang dihasilkan dikonversi menjadi bahasa yang dapat dimengerti oleh si pendengar.

Jadi, proses komunikasi dapat dibagi menjadi 4 tahapan penting, yaitu:

1. *Generation* : konversi dari „maksud' (*meaning*) menjadi bentuk linguistik (*linguistic form*) oleh si pembicara.
2. *Encoding* : konversi dari *linguistic form* ke *signal* oleh si pembicara.
3. *Decoding* : konversi dari *signal* ke *linguistic form* oleh si pendengar.
4. *Understanding* : konversi dari *linguistic form* ke meaning oleh si pendengar.

PERPUSTAKAAN FAKULTAS TEKNIK
UNIVERSITAS INDONESIA

BAB III
Text-to-Speech Synthesis

Tujuan dari *text-to-speech* (TTS) *synthesis* adalah untuk mengkonversi *input* berupa teks menjadi ucapan yang dapat dimengerti oleh pendengar. Jadi, TTS mengirimkan informasi dari mesin kepada manusia. Mengenai metode apa yang harus digunakan oleh sebuah mesin (komputer) untuk dapat melakukan konversi *text-to-speech*, dapat diketahui melalui persoalan yang terkait pada manusia ketika membaca suatu teks, kemudian melafalkannya. Berangkat dari sini, maka metode tersebut baru dapat ditentukan.

Membaca adalah proses men-*decode*-kan *signal* tulisan menjadi *message*, lalu berusaha memahami *message* tersebut untuk mendapatkan suatu pengertian (pemahaman). Membaca dengan bersuara (*reading aloud*) berarti proses mengkonversi *signal* tulisan menjadi *signal* dalam bentuk yang lain, yaitu ucapan (*speech*). Agar suatu mesin dapat melakukan proses tersebut, maka ada beberapa permasalahan yang harus diselesaikan terkait dengan proses membaca:

1. Bagaimana men-*decode*-kan *signal* tulisan
2. Apakah pesan yang dapat ditangkap dari *signal* tulisan tadi mengandung informasi yang cukup sehingga dapat dibuat ucapan yang dibutuhkan agar dapat memiliki pesan atau maksud yang sama dengan *signal* tulisan tadi
3. Ketika seseorang membaca bersuara, apa hubungan antara penulis, pembaca, dan pendengar
4. Apa yang diinginkan oleh si pendengar dari suara sintesis (*synthetic voice*).

Speech signal merupakan gelombang kontinu (gelombang akustik). *Signal* ini dibentuk dari aktifitas organ suara dalam meresponi perintah dari otak. Sedang tulisan berbeda dalam beberapa hal dengan *speech*. Pertama, tulisan adalah sesuatu yang dapat dilihat. Dahulu, tulisan dapat dilihat dalam bentuk tanda-tanda yang terdapat di atas suatu halaman atau material lainnya, dan sekarang kita sering melihat tulisan terdapat pada layar komputer atau layar lainnya. Huruf Braille tetap merupakan suatu tulisan meskipun seseorang tak perlu melihat untuk dapat membacanya, karena tulisan jenis ini memang dapat dibaca baik dengan melihat ataupun tanpa melihat. Yang menjadi pembeda utama antara tulisan

dengan *speech* adalah bahwa tulisan bersifat permanen, sedang *speech* tidak. Tak peduli tulisan tersebut ditulis dengan menggunakan pena atau hal lainnya, tulisan itu tidak hilang setelah ditulis. Seseorang tentu saja dapat menghapus tulisan tersebut setelah ditulis, namun sudah pasti bukan hal ini yang dimaksud. Idenya adalah dengan tulisan seseorang dapat merekam suatu informasi, sehingga orang tersebut dapat membacanya kembali di lain waktu atau mengirimnya ke suatu tempat.

Tulisan dapat merekam suatu bahasa yang dapat dimengerti oleh manusia dalam bentuk linguistik dari bahasa itu daripada dalam bentuk *speech signal* itu sendiri. Ketika seseorang menulis apa yang dikatakan oleh orang lain, orang tersebut tidak menulis semua informasi yang terdapat pada *speech signal* itu. Dia tidak merekam cara bicara si pembicara, ataupun cara pembicara itu berbicara, melainkan kita merekam pesan yang terkandung dalam *speech signal* tersebut yang dituangkan dalam bentuk linguistik bahasa yang bersangkutan (kata-kata).

Selain hal itu, dalam tulisan ada satu hal yang paling penting dalam tulisan itu yang diabaikan, yaitu prosodi. Prosodi berfungsi untuk membentuk emosi dari pembicara. Dalam bahasa tulisan, prosodi ini memiliki komponen yang terbatas karena tidak adanya cara untuk mengekspresikan beberapa fitur di dalam prosodi tersebut. Kita memang dapat memberikan tanda baca seperti tanda seru, tanda tanya, dan sebagainya untuk memberikan penekanan, namun hal-hal seperti kemarahan, kesakitan, atau sarkasme, tidak ada cara yang khusus untuk mengutarakannnya.

Dalam tulisan terkandung bahasa yang bukan merupakan bagian dari linguistik, misalnya bahasa matematika. Ketika menemui suatu bahasa matematika, mesin harus dapat membedakannya dengan bahasa natural (*natural language*). Kita ambil sebagai contoh, $x^2 + y^2$. Kita tahu bahwa angka 2 di atas huruf x dan y menyatakan pemangkatan, sehingga kalimat ini dibaca sebagai ,,eks kuadrat ditambah y kuadrat'.

Permasalahan tersebut termasuk di dalam masalah semiotik. Bahasa natural dikenal sebagai sistem semiotik. Mesin harus menginterpretasikan kalimat yang mengandung lambang semiotik ketika menemui kalimat yang mengandung lambang tersebut. Misalnya:

- $x^2 + y^2 = z^2$
- Saya lahir pada 12/12/12
- Berat barang itu 23 kg
- Saya akan mengirimnya ke paul@yahoo.com
- Dan seterusnya.

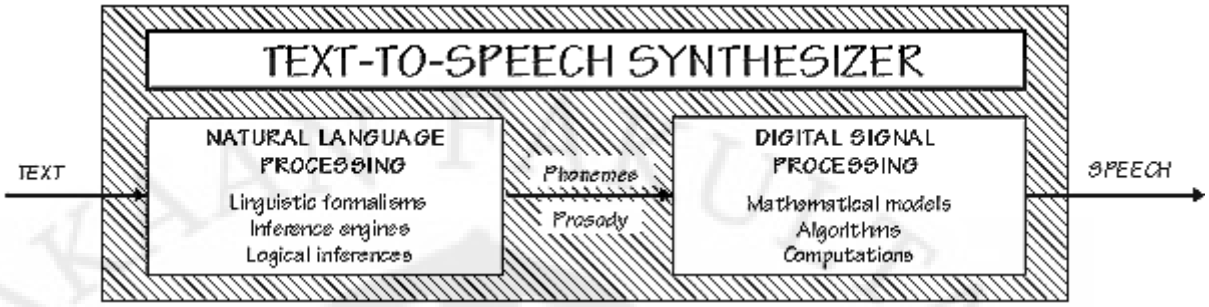
Agar mesin dapat memroses tulisan seperti itu, mesin harus dapat mengidentifikasi bagian mana dari teks yang berhubungan dengan masalah semiotik.

Masalah-masalah yang dibahas di atas merupakan beberapa masalah yang sering ditemui di dalam proses membaca. Berangkat dari pengertian kita akan hal cara membaca, kita dapat mendefinisikan model dari TTS yang mampu melakukan proses yang kita inginkan.

Metodologi yang digunakan dalam TTS adalah dengan menggunakan representasi dari akustik ucapan untuk sintesis bersama-sama dengan penganalisisa tata-bahasa teks untuk menghasilkan pelafalan yang benar (berupa 'isi' teks, apa yang diucapkan) dan prosodi („melodi' sebuah kalimat, bagaimana kalimat tersebut diucapkan). Sistem sintesis umumnya ditentukan oleh 3 karakteristik:

1. Keakuratan (*accuracy*) : apakah sistem TTS melafalkan teks-teks seperti: akronim, nama, URL, alamat e-mail, dan sebagainya dengan tepat?
2. Intellijibilitas (*intelligibility*) : apakah teks yang dilafalkan tersebut dapat dimengerti oleh pedengar (dengan kata lain, sejauh mana kemampuan mesin tersebut untuk memberikan kemungkinan bagi pendengar untuk men-*decode*-kan *speech* yang dihasilkan secara tepat)?
3. Kealamian (*naturalness*) : apakah ucapan yang dihasilkan seperti yang diucapkan oleh manusia?

TTS *synthesizer* terdiri dari 2 modul utama, yaitu *Natural Language Processing* (NLP) dan *Digital Signal Processing* (DSP) seperti digambarkan oleh Gambar 3.1:

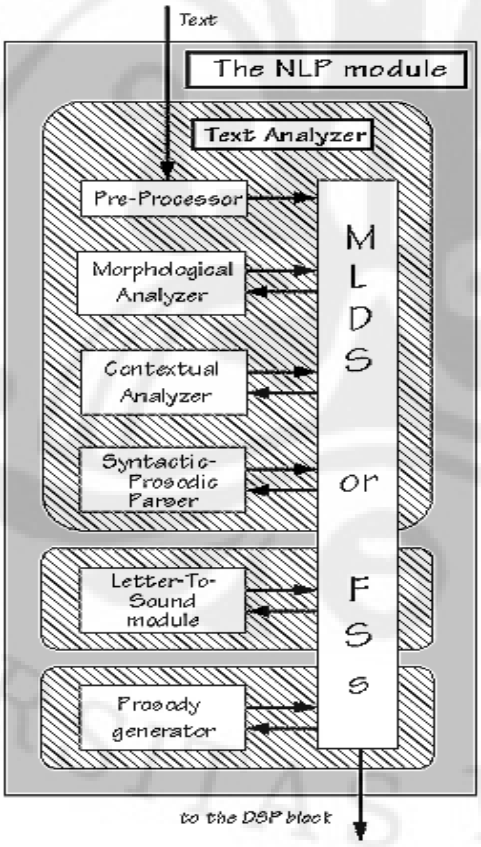


Gambar 3.1. Diagram sederhana sistem TTS [7].

Sebagai alat untuk membaca, TTS *synthesizer* ini memiliki modul NLP untuk menghasilkan suatu transkrip fonetik dari hasil membaca berikut dengan prosodi dari teks tersebut. Modul DSP berfungsi untuk memroses *input* dari NLP, yaitu berupa fonem dan prosodi tadi sehingga dapat dihasilkan ucapan (*speech*).

3.1. Komponen NLP (Natural Language Processing)

Kerangka dasar dari NLP dapat dilihat pada Gambar 3.2.



Gambar 3.2. Modul NLP dari sistem TTS [7].

Dari Gambar 3.2, dapat lihat bahwa modul NLP terdiri dari 3 bagian utama, yaitu ***Text Analyzer, Letter-to-Sound Module,* dan *Prosody Generator*.** Agar dapat menghasilkan *letter-to-sound* berikut dengan prosodinya, maka modul NLP haruslah memiliki blok penganalisis *morpho-syntactic analyser*. Adalah diperlukan untuk dapat membagi kalimat menjadi beberapa bagian, seperti klausa, lalu akhirnya menjadi kata (*part-of-speech*). Setidaknya ada 2 alasan mengapa demikian:

1. Transkrip fonetik yang akurat hanya dapat diperoleh bila *part-of-speech* dari suatu kata tersedia. Dan juga hubungan antar-kata-kata tersebut diketahui.
2. Prosodi yang alami tergantung pada sintax dari suatu kalimat.

3.1.1. Text Analyzer

Komponen ini bertugas untuk menentukan struktur dokumen, pengkonversian simbol nonortografi, dan memecah (*parsing*) struktur bahasa dan maknanya. Blok ini terdiri dari beberapa komponen berikut:

1. *Pre-processing module*

   Modul ini bertugas untuk mengatur kalimat yang masuk agar dapat menjadi susunan kata-kata yang dapat diatur. Modul ini mengidentifikasi angka, singkatan, akronim, dan idiom lalu mengubahnya menjadi suatu full teks ketika diperlukan. Masalah yang tersulit yang dihadapi oleh modul ini adalah ketika modul ini menemui tanda baca (titik, koma, tanda seru, tanda tanya, dan sebagainya). Hal ini dikarenakan sifat ambiguitas yang dimiliki oleh tanda baca tersebut. Misalnya tanda **titik (.)**. Tanda titik ini dapat berarti akhir dari suatu kalimat, namun juga dapat berarti tanda dari suatu singkatan kata, atau bagian dari alamat suatu web, atau yang lainnya.

2. *Morphological Analysis module*

   Modul ini bertugas untuk menebak semua kemungkinan cara pelafalan dari kata-kata yang masuk. Hal ini dilakukan dengan cara memecah kata yang masuk menjadi unit-unit grafem-nya.

3. *Contextual Analysis module*

   Modul ini bertugas untuk menyesuaikan makna kata dalam konteks kalimat sehingga dapat ditentukan apakah kata itu berupa kata sifat, kata kerja, kata

benda, dan sebagainya (*part-of-speech*). Hal ini dapat dilakukan dengan beberapa metode, seperti *n-grams*, dengan *multi-layer perceptron* (neural network), atau dengan *local, non-stochastic grammars*.

4. *Syntactic Prosody Parser*

   Modul ini bertugas untuk menentukan struktur teks (klausa atau frasa) sehingga dapat ditentukan prosodi yang tepat.

3.1.2. Letter to-Sound Module (Automatic Phonetization)

Modul *Letter-to-Sound* ini bertugas untuk menentukan bagaimana cara pengucapan teks yang masuk (penentuan fonetik). Untuk dapat melakukan tugas tersebut, ada 2 cara pada umumnya yang dapat dilakukan:

1. *Dictionary-based strategy*
2. *Rule-based strategy.*

3.1.2.1. Dictionary-based Strategy

Metode ini dilakukan dengan cara menyimpan pengetahuan mengenai fonetik sebanyak mungkin ke dalam sebuah leksikon (daftar istilah dalam suatu bidang disusun menurut abjad dan dilengkapi dengan keterangannya). Agar dapat menjaganya tetap berukuran kecil, isi dari leksikon tersebut umumnya dibatasi hanya pada morfem-morfem kata yang masuk. Morfem yang tak dapat ditemukan di dalam leksikon akan ditentukan melalui aturan-aturan. Setelah fonemik tiap kata telah ditentukan, dilakukan pro-processing fonetik dengan tujuan menghaluskan kalimat.

3.1.2.2. Ruled-based Strategy

Metode ini dilakukan dengan aturan *letter-to-sound* (*grapheme-to-phoneme*) untuk kata-kata yang tidak dapat diprediksi pelafalannya dimana tidak terdapat dalam leksikon yang telah dibuat. Kata-kata tersebut dimasukkan ke dalam *exception dictionary*. Umumnya, sistem TTS memerlukan ratusan, bahkan ribuan aturan untuk dapat mencakup semua kata-kata yang tidak terdapat di dalam leksikon. Dan perlu diketahui bahwa banyak sekali kata-kata yang perlu

dilafalkan dengan cara-cara yang khusus. Bahkan jumlah dari kata-kata ini dapat melebihi kata-kata yang dapat diprediksi pelafalannya.

Metode ini menggunakan sekelompok aturan yang kemudian diterapkan pada kata yang tidak diketahui untuk menemukan cara pelafalannya. Pendekatan yang paling umum adalah dengan memroses karakter dari kiri ke kanan lalu untuk tiap karakter diterapkan satu atau dua aturan guna menghasilkan fonemnya. Pendekatan tersebut adalah melalui bentuk:

$$\boldsymbol{A \text{ -> } B \,/\, X\_Y} \qquad \textbf{(3.1)}$$

yang dibaca "*A* dibaca sebagai B bila A diantara X dan Y.*"*

3.1.3. Prosody Generator

Prosodi adalah nada (tinggi-rendah), tekanan (keras-lemah), durasi (panjang-pendek), jeda (kesenyapan), dan intonasi dalam bahasa. Prosodi memiliki fungsi yang khusus di dalam komunikasi, seperti untuk membedakan maksud dan makna dalam tataran kalimat (sintaksis) [8].

Masalah nada (*pitch*) dalam penuturan bahasa Indonesia tidak fungsional atau tidak membedakan makna. Tapi akan memberikan ekspresi perasaan yang berbeda ketika kita mengucapkan suatu kata atau kalimat dengan nada yang berbeda. Ketika penutur mengucapkan [aku], [membaca], [buku] dengan nada tinggi, sedang, atau rendah, maknanya sama saja. Begitu juga dalam tingkatan lingual yang lebih besar, seperti: frase, klausa, dan kalimat. Bahkan, penuturan yang diucapkan secara berlagu (bernyanyi) pun maknanya sama dengan ketika diucapkan secara biasa.

Namun, lain masalahnya dengan bahasa yang lain, misalnya bahasa Mandarin, dimana nada amat berperan penting dalam membedakan makna. Dalam bahasa Mandarin, tiap kata memiliki 4 macam nada, yaitu mendatar (-), menukik ke bawah (\), menukik ke bawa dan ke atas (v), dan menukik ke atas (/). Dan makna yang dihasilkannya pun berbeda untuk tiap nada.

Berbeda dengan nada, tekanan dalam bahasa Indonesia berfungsi membedakan maksud dalam tataran kalimat (sintaksis), tapi tidak berfungsi membedakan makna dalam tataran kata (leksis) [8]. Misalnya pada kalimat

„Kemarin teman saya meminjam uang di bank'. Kalimat ini bisa diucapkan dengan 6 kemungkinan variasi tekanan sebagai berikut:

1. ***Kemarin*** *teman saya meminjam uang di bank*
2. *Kemarin* ***teman*** *saya meminjam uang di bank*
3. *Kemarin teman* ***saya*** *meminjam uang di bank*
4. *Kemarin teman saya* ***meminjam*** *uang di bank*
5. *Kemarin teman saya meminjam* ***uang*** *di bank*
6. *Kemarin teman saya meminjam uang* ***di bank***

Kalimat (1) mendapat tekanan pada kata *kemarin*. Maksudnya adalah „teman saya meminjam uang di bank *kemarin,* bukan *sekarang* atau waktu lain'. Kalimat (2) mendapat tekanan pada kata *teman.* Maksudnya adalah „yang kemarin meminjam uang di bank itu adalah *teman saya,* bukan *saudara saya* atau *yang lain'*. Kalimat (3) mendapat tekanan pada kata *saya.* Maksudnya adalah „memang teman *saya* yang kemarin meminjam uang di bank, bukan teman *kamu* atau *yang lain'*. Kalimat (4) mendapat tekanan pada kata *meminjam.* Maksudnya adalah „teman saya kemarin memang *meminjam* uang di bank, bukan menabung atau menukar, atau yang lain.' Kalimat (5) mendapat tekanan pada kata *uang.* Maksudnya adalah „yang dipinjam oleh teman saya di bank kemarin itu adalah *uang*, bukan *emas* bukan *barang berharga lain'*. Kalimat (6) mendapat tekanan pada kata *di bank*. Maksudnya adalah „kemarin teman saya memang meminjam uang *di bank*, bukan *di koperasi, di rumah,* atau *di tempat lainnya'*.

Durasi atau panjang-pendek juga tidak fungsional (tidak membedakan makna) dalam tataran kata, tapi fungsional dalam tataran kalimat. Jadi, masalah durasi memberikan makna yang berbeda untuk durasi yang berbeda ketika mengucapkan suatu kalimat. Misalnya, perhatikanlah kalimat berikut:

- *Awas, jatuh!* Diucapkan [awa:s / jatu:h]
  Kalimat ini memiliki durasi yang lebih lama pada silaba terkhir. Dan ini berakibat memberikan makna „mencari perhatian' atau „penyangatan'.
- *Satu, dua, tiga!* Diucapkan [satu: / dua: / tiga:].
  Kalimat ini memberikan makna „aba-aba'.

Jeda atau kesenyapan terjadi di antara 2 bentuk linguistik, baik antarkalimat, antarfrase, antarkata, antarmorfem, antarsilaba, maupun antarfonem. Dalam bahasa Indonesia, jeda lebih fungsional bila disbanding dengan jenis prosodi yang lain. Misal pada kalimat berikut:

1. ***Anak / pejabat yang nakal*** *itu telah dimejahijaukan.*
2. ***Anak pejabat / yang nakal*** *itu telah dimejahijaukan.*

Dengan perbedaan jeda yang agak lama antara *anak* dan *pejabat* (kalimat 1) dan antara *pejabat* dan *yang (*kalimat 2) makna kalimat itu berbeda. Pada kalimat 1, ‚yang nakal adalah pejabat', sedang pada kalimat 2, ‚yang nakal adalah anak pejabat'.

Intonasi dalam bahasa Indonesia sangat berperan penting dalam pembedaan maksud kalimat. Bahkan, dengan dasar kajian pola-pola intonasi ini, kalimat bahasa Indonesia dibedakan menjadi kalimat berita (deklaratif), kalimat tanya (interogatif), dan kalimat perintah (imperatif).

Seperti yang telah kita lihat di atas bahwa membentuk prosodi dengan mempertimbangkan faktor-faktor di atas adalah sulit. Oleh karena itu, seringkali sistem TTS memiliki *output* suara yang netral, tidak memihak pada makna dan maksud tertentu.

Untuk dapat mengetahui informasi prosodi dari suatu teks, kita dapat melakukannya dengan menggunakan metode tertentu. Salah satu metode yang paling ampuh untuk melakukan proses ini adalah ToBI (*Tone and Break Indices*).

ToBI merupakan salah satu metode atau cara untuk melakukan labelisasi terhadap kejadian-kejadian prosodi yang terdapat di dalam pembacaan suatu kalimat. Jadi, metode ini memungkinkan kita untuk dapat mengetahui secara tertulis akan prosodi yang terkandung di dalam tiap kata dalam suatu kalimat. Yang dimaksud dengan kejadian prosodi di sini mencakup dua kategori [14], yaitu:

1. Peristiwa adanya penekanan terhadap suatu suku kata atau kata tertentu pada suatu kata atau kalimat yang lebih dibandingkan dengan suku kata atau kata yang lain.
2. Peristiwa pengelompokan (*phrasing* atau *grouping*) pada sederet suku kata.

Hal ini berkaitan dengan cara pembacaan antara suku kata yang satu dengan suku kata berikutnya atau antara kata yang satu dengan kata berikutnya. Apakah suku kata-suku kata atau kata-kata tersebut harus dibaca secara kontinu (langsung tanpa ada jeda) atau harus diberi jeda, baik itu sedikit atau banyak.

Misalnya pada kalimat: "**Kemarin**, teman saya Budi meminjam uang di Bank." Kata kemarin yang di-*bold* dimaksudkan untuk menandakan bahwa kata kemarin ini diberi tekanan yang lebih dibandingkan dengan kata yang lain sehingga memberi pengertian kepada kita bahwa si Budi meminjam uang **kemarin**, **bukan besok atau lusa.** Masalah ini berkaitan dengan kategori kejadian prosodi yang pertama.

Kemudian, antara kata kemarin dengan kata teman dipisahkan dengan tanda baca koma. Hal ini memberi pengertian kepada kita bahwa antara kata kemarin dengan kata teman itu harus diberi jeda. Masalah ini berkaitan dengan kategori kejadian prosodi yang kedua.

Lalu, dari parameter apakah yang dapat dilakukan pengukuran untuk dapat menentukan penekanan dan pengelompokan ini? Jawabannya ada beberapa macam. Namun parameter yang paling sering digunakan adalah pola f0 (berkaitan dengan pola *pitch* suatu ucapan) dan durasi. Sedang parameter-parameter akustik yang lain seperti amplitudo (yang berkaitan dengan kekerasan suara), kualitas suara, *pausing*, kekuatan artilulasi, dan sebagainya, sedang masih dalam tahap penelitian.

Labelisasi dengan metode ToBI ini dilakukan dengan pelabelan pada 4 lapisan [14]:

1. Lapisan Tone, untuk melakukan labelisasi nada.
2. Lapisan Ortografik, untuk melakukan labelisasi kata.
3. Lapisan indeks penjedaan (Break-Index), untuk melakukan labelisasi penjedaan.
4. Lapisan tambahan, untuk melakukan labelisasi terhadap informasi tambahan seperti apakah kata atau suku kata tersebut sedang diucapkan dalam kondisi batuk atau tertawa, dan sebagainya.

Untuk pelabelan *tone*, ada beragam nada yang dapat diaplikasikan terhadap suatu kalimat. Secara garis besar, terdapat 2 macam nada yang didefinisikan di sini, yaitu H (*High*, merepresentasikan *pitch* yang tinggi) dan L (*Low*, merepresentasikan *pitch* yang rendah). Kemudian gabungan dari kedua nada tersebut dan tambahan atribut yang lain membuat banyak macam nada yang dihasilkan. Diantaranya [14]:

1. H* : *High pitch accent*
2. L* : *Low pitch accent*
3. L+H* : *Bitonal pitch accent with low tone followed by a high tone prominence*
4. L*+H : *Bitonal pitch accent with low tone prominence followed by a high tone*
5. !H* : *Downstepped High pitch accent*
6. L+!H* : *Bitonal pitch accent with low tone followed by a downstepped high tone prominence*
7. L*+!H : *Bitonal pitch accent with low tone prominence followed by a downstepped high tone*
8. H+!H* : *Bitonal pitch accent with high tone followed by a downstepped high tone prominence*
9. L-L% : *Low phrase tone, low boundary tone*
10. H-H% : *High phrase tone, high boundary tone*
11. L-H% : *Low phrase tone, high boundary tone*
12. H-L% : *High phrase tone, low boundary tone*
13. H- : *High phrase tone*
14. L- : *Low phrase tone*

Namun, nada yang dipakai di dalam skripsi ini hanya ada 6 buah, yaitu L, H*, L-L%, H-H%,L-H%, dan H-L%. Hal ini dilakukan mengingat bahwa kejadian prosodi yang dibuat di dalam modul ProsodyPredicting hanyalah sebatas per-kata saja, tidak per-kalimat. Akibatnya, kejadian prosodi yang dapat diprediksi pada suatu kalimat tidak dapat dilakukan secara lengkap. Jadi, masalah mengenai apakah suatu kata tertentu yang terdapat di dalam suatu kalimat

mengalami penekanan maksud dibandingkan dengan kata lainnya tidak dapat ditentukan.

Kejadian prosodi L maksudnya adalah suatu kata atau suku kata memiliki *pitch* yang rendah. Namun rendah atau tingginya *pitch* dari suatu suara tidak dapat langsung ditetapkan begitu saja, sebab ada orang yang memang dari asalnya atau secara alamiahnya memang memiliki suara ber-*pitch* tinggi. Bila kasusnya demikian, maka kejadian prosodi yang demikian adalah L.

Kejadian prosodi H* maksudnya adalah suatu kata atau suku kata memiliki *pitch* yang paling tinggi dibandingkan dengan *pitch* dari kata atau suku kata yang lain.

Kejadian prosodi L-L% memiliki cara pembacaan yang agak cepat pada antar suku kata (*low phrase accent*) dan memiliki *pitch* yang rendah (*low boundary tone*) pada akhir pembacaan suatu kata. Sehingga kejadian prosodi ini cocok untuk suatu kalimat berita yang diakhiri dengan tanda titik (.).

Kejadian prosodi H-H% memiliki cara pembacaan yang agak lambat pada antar suku kata (*high phrase accent*) dan memiliki *pitch* yang tinggi pada akhir pembacaan suatu kata (high boundary tone). Kejadian prosodi ini cocok untuk suatu kalimat yang diakhiri dengan tanda tanya (kalimat tanya).

Kejadian prosodi L-H% memiliki cara pembacaan yang agak cepat pada antar suku kata (*low phrase accent*) dan memiliki *pitch* yang tinggi pada akhir pembacaan suatu kata (high boundary tone). Kejadian prosodi ini juga cocok untuk kata yang diikuti dengan tanda tanya.

Kejadian prosodi H-L% memiliki cara pembacaan yang agak lambat pada antar suku kata (*high phrase accent*) dan memiliki *pitch* yang rendah pada akhir pembacaan suatu kata (*low boundary tone*). Kejadian prosodi ini cocok untuk kata yang diikuti dengan koma (,).

Break Index menandakan tingkat atau level dari penjedaan antara dua kata (atau suku kata, dalam skripsi ini). Terdapat 5 jenis indeks penjedaan (*break index*) yang didefinisikan:

0: Indeks ini menandakan bahwa antara kata yang satu dengan kata berikutnya atau antara suku kata yang satu dengan suku kata berikutnya tidak memiliki penjedaan sama sekali.

1: Indeks ini menandakan penjedaan yang biasa terdapat pada jeda antar kata.

2 dan 3: Dua indeks ini disebut juga sebagai *intermediate phrasing*. Contoh dari pemakaian indeks ini terdapat pada ketika kita sedang melakukan aba-aba: "satu, dua, tiga!", misalnya. Indeks jenis ini tidak terlalu dibahas di dalam skripsi ini.

4: Indeks ini biasanya menandakan akhir dari suatu kalimat.

Prosodi yang dapat ditentukan untuk suatu kata dilihat dari apakah akhir dari kata tersebut mengandung tanda baca seperti: koma (,), titik (.), tanda tanya (?), tanda seru (!) atau tidak ada tanda baca sama sekali (). Dari atribut tanda baca tersebut, maka aturan kejadian prosodi yang dibuat di dalam skripsi ini adalah sebagai berikut:

1. Bila akhir dari suatu kata tidak memiliki tanda baca sama sekali, maka kejadian prosodi yang didefinisikan adalah L, terlepas dari apakah nantinya suara yang dihasilkan akan memiliki *pitch* yang tinggi atau rendah.
2. Bila akhir dari suatu kata memiliki tanda baca koma (,), maka kejadian prosodi yang didefinisikan adalah H-L%.
3. Bila akhir dari suatu kata memiliki tanda baca titik (.), maka kejadian prosodi yang didefinisikan adalah L-L%.
4. Bila akhir dari suatu kata memiliki tanda baca tanda Tanya (?), maka kejadian prosodi yang didefinisikan adalah H-H%, meskipun kejadian prosodi L-H% juga memungkinkan. Namun, terlepas dari apakah nantinya suara yang dihasilkan akan memiliki kejadian prosodi L-H% atau H-H%, kejadian prosodi untuk kasus ini tetap didefinisikan sebagai H-H%.
5. Bila akhir dari suatu kata memiliki tanda baca tanda seru (!), maka kejadian prosodi yang didefinisikan adalah H*.

3.2. Komponen DSP (*Digital Signal Processing*)

Secara intuitif, operasi yang dilakukan oleh modul DSP (*Digital Signal Processing*) ini mirip dengan pekerjaan komputer untuk mengendalikan otot *articulatory* dan frekuensi getaran *vocal cord* sehingga *signal* keluaran yang dihasilkan akan sesuai dengan *input*. Untuk dapat melakukan hal ini, modul DSP

harus memperhitungkan hambatan-hambatan dalam pengucapan atau pelafalan. Dan hal tersebut dapat dilakukan dengan 2 cara:

1. Secara eksplisit, membentuk aturan-aturan (*a series of rules*) yang menggambarkan pengaruh fonem antar-kata.
2. Secara implisit, menyimpan sejumlah contoh transisi fonetik dan pelafalan ke dalam suatu *database*, lalu menggunakan *database* tersebut sebagaimana adanya mereka sebagai alat akustik.

Dua kelas utama dari sistem TTS yang muncul untuk mengerjakan alternatif tersebut adalah *synthesis-by-rule* dan *synthesis-by-concatenation.*

3.2.1. Rule-based synthesizer

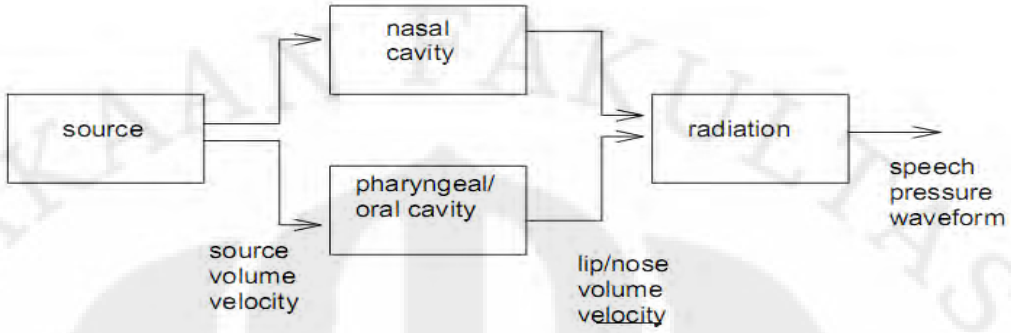
Sintesa jenis ini banyak dipakai oleh para *phoneticians* dan *phonologist*. *Rule-based synthesizer* ini selalu muncul dalam bentuk *formant synthesizer* . Dan sering disebut sebagai teknik penyintesaan generasi pertama (*first generation*). *Formant synthesizer* paling berhubungan dengan frekuensi *formant* dan anti-*formant* dan *bandwidth*-nya beserta dengan gelombang glottal (*glottal waveform*). Namun, frekuensi *formant* dan *bandwidth*-nya sulit untuk diestimasi dari data ucapan. Oleh karena itu dibutuhkan banyak *trial and error* untuk dapat mengatasi masalah error.

Formant adalah puncak dari spektrum frekuensi suara yang dihasilkan oleh resonansi akustik. Dalam fonetik, *formant* mengacu kepada suara yang dihasilkan oleh *vocal track*. Dalam bidang akustik, *formant* mengacu kepada resonansi dalam sumber bunyi [10].

Meskipun demikian, *rule-based synthesizer* merupakan pendekatan yang amat baik dalam *speech synthesis*. Metode ini memungkinkan untuk mempelajari karakteristik dari suara *speaker-dependent* sehingga pengalihan dari satu *synthetic voice* ke *synthetic voice* yang lain dapat dilakukan dengan bantuan aturan-aturan yang khusus dalam *database*.

Formant sintesis mengadopsi suatu pendekatan **modular, *model-based*, *acoustic-phonetic*** terhadap permasalahan sintesis. *Formant synthesizer* menggunakan model *acoustic tube* dengan cara sedemikian sehingga pengontrolan terhadap elemen *acoustic tube* dapat dengan mudah terhubung

dengan properti *acoustic-phonetic*. Blok diagram dasar dari *Formant synthesizer* ini dapat dilihat pada Gambar 3.3.

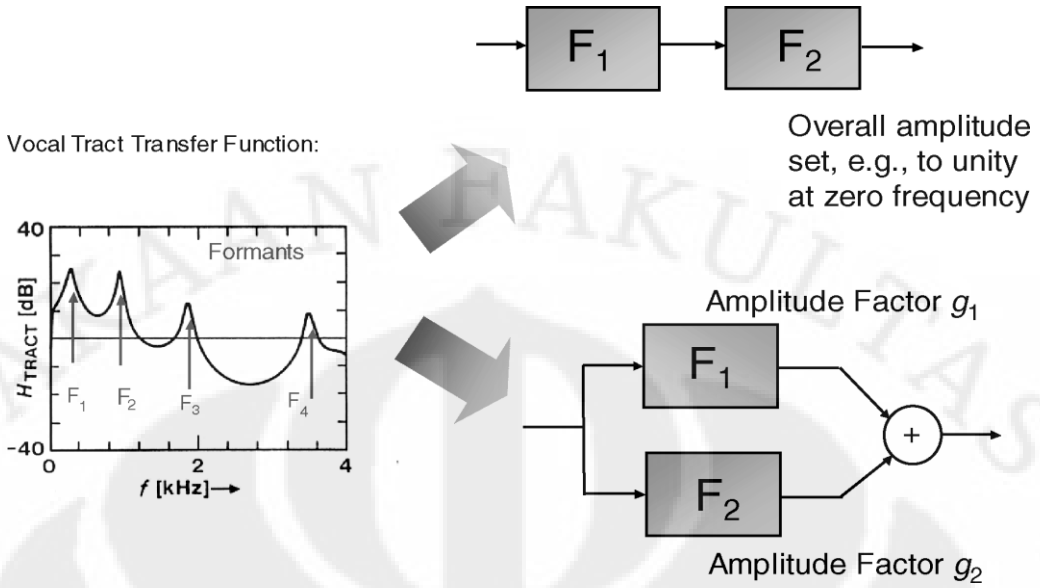


Gambar 3.3. Blok diagram *Formant synthesizer* sederhana [6].

Dari diagram blok di Gambar 3.3 di atas, suara dihasilkan dari suatu sumber, baik yang bersifat *voiced sounds* maupun *white noise* untuk *obstruent sound*. *Signal* dari sumber suara ini diumpankan ke model *vocal track* yang direpresentasikan sebagai *oral cavities* dan *nasal cavities*. Oral dan *nasal cavities* ini disusun secara parallel. Sehingga *signal* dari sumber dapat masuk ke *oral cavities* maupun ke *nasal cavities* untuk menghasilkan suara nasal. Terakhir, keluaran dari model *vocal track* ini dikombinasikan dan melewati komponen *radiation* yang mensimulasikan karakteristik beban dan propagasi bibir dan hidung.

Sumber suara untuk *vowel* dan *voiced consonants* dapat dihasilkan dari suatu dungsi periodic domain waktu atau melalui sederet impuls yang diumpankan ke filter LTI (*Linear Time Invariant) glottal.*

Baik *nasal cavities* maupun *oral cavities* merupakan suatu resonator. Dan resonator ini merupakan filter orde 2 yang disusun baik secara kaskade / seri ataupun parallel. Gambar 3.4 di bawah ini menggambarkan *formant synthesizer* berikut dengan susunan resonatornya.



Gambar 3.4. *Formant synthesizer* dalam susunan seri (atas) dan parallel (bawah). Fungsi alih dari *vocal track* (kiri) menunjukkan 4 buah *formant* dalam rentang frekuensi dari 0 – 4kHz. Fungsi alih ini dapat diaproksimasi dengan menggunakan filter orde 2 yang disusun baik secara seri maupun paralel [9].

Tugas dari filter tersebut adalah untuk mengaproksimasi semua resonansi *vocal track* (*formant*) yang bermula dari fungsi alih *vocal tract* yang ada di sebelah kiri Gambar 3.4 yang menghubungkan antara aliran volume pada bibir (*output*) dengan aliran volume pada *glottis* (*input*). Filter yang disusun secara seri bekerja dengan baik dalam mengaproksimasi *non-nasal vocal track*. Dengan menggunakan pendekatan ini, kita hanya perlu untuk menentukan frekuensi *formant*, *bandwidth*-nya, serta *gain factor*. Sedang filter yang disusun secara parallel dapat mengaproksimasi *nasal sound* dan dapat juga berlaku untuk *voiced / unvoiced sound*.

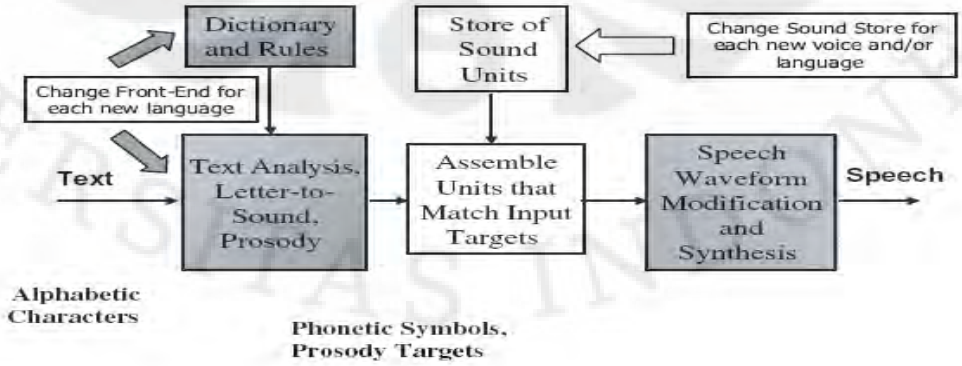
Menurunkan aturan untuk menyintesa ucapan yang sedang berlangsung adalah masalah utama dalam *formant synthesis*. Aturan-aturan tersebut adalah menentukan pewaktuan (*timing*) dari sumber ucapan (baik *voiced* maupun *non-voiced*) dan nilai dinamis (*dynamic values*) untuk semua parameter filter. Hal ini sulit dilakukan secara manual, bahkan untuk kata yang sederhana sekalipun. Penurunan secara automatic (*automatic derivation*) dapat dilakukan dengan pendekatan *analysis-by-synthesis* dengan meniru masukan berupa ucapan.

Formant synthesizer membutuhkan perlengkapan penghitungan yang cukup memadai. Beberapa sistem TTS membutuhkan memory sebesar 2 MB atau kurang. Dengan demikian, program terintegrasi (*embedded application*) seperti *talking dictionary*, kalender, dsb, atau bahkan di dalam telepon genggam (untuk membaca nama-nama dan penekanan tombol sehingga pengemudi tidak perlu mengalihkan mata mereka ketika mengemudi) dapat dimungkinkan dengan menggunakan *formant synthesis*. Kualitas suara dapat ditingkatkan, namun biasanya mustahil untuk dapat mencocokkan dengan kualitas suara yang mirip dengan sebenarnya.

3.2.2. *Concatenative Synthesizer*

Berlawanan dengan *rule-based synthesizer* , *concatenative synthesizer* memiliki pengetahuan yang sangat terbatas mengenai data yang ditangani. Metode ini dilakukan dengan cara menggabungkan (*concatenate*) segmen-segmen ucapan yang telah direkam sebelumnya. Umumnya metode ini memang menghasilkan *output* sintesa ucapan yang natural (mendekati ucapan manusia). Namun dari segi *intelligibility* (sejauh mana *output* ucapan tersebut dapat dimengerti oleh manusia), metode ini masih memiliki kekurangan.

Rekaman dari segmen-segmen ucapan tersebut dapat disimpan di dalam *database* dalam bentuk gelombang (*uncoded*) atau dalam bentuk *encoded* disesuaikan dengan metode pengkodingan ucapan (*speech coding method*) yang dipakai. Gambar blok diagram dari c*oncatenative synthesizer* dapat dilihat pada Gambar 3.5.



Gambar 3.5. Blok diagram *concatenative* TTS [9].

Pada Gambar 3.5 di atas, bagian f*ront-end* ditunjukkan pada bagian sebelah kiri Gambar 3.5. Sedang bagian *back-end* ditunjukkan oleh bagian sebelah kanan Gambar 3.5. Bagian *front-end* bertugas untuk mengkonversi *input* berupa *string* dari teks menjadi string dari simbol fonetik dan prosodi. Bagian *front-end* menggunakan kumpulan dari aturan-aturan dan atau kamus pelafalan (*pronunciation dictionary*). Bersama dengan string dari simbol fonetik, metode ini menghasilkan *pitch*, durasi fonem, dan amplitudo dari tujuan yang diinginkan. Dan oleh blok bagian tengah dari Gambar 3.5, keluaran dari front-end akan dicocokkan dengan rekaman segmen suara yang telah disimpan sebelumnya dalam suatu *database* dan rekaman yang sesuai akan dipilih untuk diproses oleh blok diagram berikutnya.

Menyimpan semua unit kata adalah tidak dapat dipraktekkan untuk sistem TTS karena banyaknya kata-kata (sampai ratusan ribu) yang harus dibaca oleh seseorang untuk direkam. Bahkan, meskipun berhasil direkam dalam waktu beberapa minggu, ketidaksempurnaan dari perekaman fonetik dan koartikulasi (*coarticulation*) akan menyebabkan suara ucapan yang tidak *natural*. Pada sisi lainnya, menggunakan suara ucapan (sekitar 50 untuk bahasa Inggris) juga tidak menghasilkan keluaran yang memuaskan karena besarnya efek koartikulasi (*coarticulatory effect*) yang muncul pada fonem yang bersebelahan. Oleh karena itu, transisi dari suatu unit ke unit berikutnya akan terdengar kecacatan (*glitches*) yang mengganggu seseorang untuk dapat mengerti kata yang dihasilkan.

Implementasi TTS sampai pertengahan tahun 1990-an menggunakan satu dari 2 tipe unit penyimpanan, *diphone* dan *demisyllable. Diphone* adalah suatu potongan kecil dari ucapan dari pertengahan ucapan yang satu ke pertengahan ucapan yang lain. Pertengahan dari suatu ucapan cenderung merupakan daerah yang paling stabil secara akustik. Jadi, *diphone* merepresentasikan transisi akustik dari daerah pertengahan (*midsection*) ucapan yang satu ke midsection ucapan berikutnya. *Diphone synthesis* memberikan intellijibilitas (*intelligibility*) yang tinggi. Kekurangan dari *diphone synthesis* ini adalah koartikulasi hanya dapat disuplai dengan fonem yang mendahului dan fonem yang berikutnya, padahal beberapa fonem dapat mempengaruhi artikulasi dari beberapa fonem. *Demisyllables* menjadi alternative *concatenative synthesis*. *Demisyllable* tercakup

di dalam satu setengah suku-kata (*syllable*). Karena unit *demisyllables* biasanya lebih panjang dibandingkan dengan *diphone*, dan memungkinkan untuk mendapatkan efek koartikulasi yang lebih baik dibandingkan dengan *diphone*, maka *demisyllables synthesis* ini seharusnya mengalami masalah *concatenation* yang lebih sedikit.

Ada beberapa syarat yang harus dipenuhi agar suatu *signal* ucapan (*speech signal*) yang dihasilkan dalam *concatenation synthesis* dapat dikatakan baik, yaitu:

1. *Signal* ucapan dapat disimpan dalam bentuk pengkompresan yang baik sehingga dapat dihasilkan *database* yang berisi rekaman segmen ucapan yang banyak. *Coder* dan *decode*r juga haruslah memiliki kompleksitas perhitungan yang rendah.
2. *Coding* dan *decoding* dapat diinterpretasikan secara jelas.
3. Algoritma dari *coding* yang digunakan harus memungkinkan „*random access'*.
4. Representasi dari ucapan yang ideal harus memungkinkan untuk dapat memodifikasi *pitch*, durasi, dan amplitudo. Hal ini penting khususnya untuk tempat penyimpanan yang sedikit menampung contoh tiap unit fonem. Sayangnya, pengalaman menunjukkan bahwa untuk kebanyakan algoritma *signal processing*, dengan memodifikasi beberapa persen dari *pitch* dapat menghancurkan tingkat kealamian (*naturalness*). Hal ini disebabkan karena memodifikasi *signal* ucapan yang telah terekam adalah jauh berbeda dengan mengatur *pitch signal* ucapan yang dapat diatur *pitch*-nya secara langsung oleh si pembicara .
5. Untuk beberapa pemakaian yang lebih *advanced*, diperlukan untuk dapat „me-melodikan' ucapan, misalnya dengan menambah aspirasi (artikulasi yang melibatkan hembusan napas), kelembutan, atau keteriakan.

Dengan syarat-syarat di atas, ada 3 kelas algoritma *speech signal processing* yang dapat digunakan, yaitu TD-PSOLA (*Time Domain Pitch-Synchronous Overlap Add*), LPC-based Algorithm (LPC = *Linear Predictive Coding*), dan *Frequency domain-based speech representation.*

Pemodelan gelombang ucapan dimodelkan sebagai susunan dari *frame-frame* kecil ucapan tersebut. Tiap *frame* tersebut dianggap sebagai sistem yang

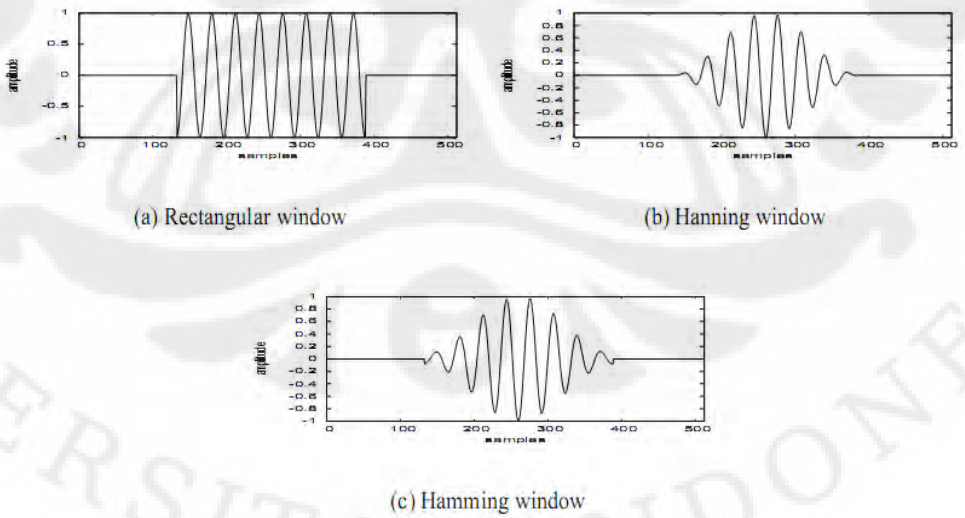
time-invariant (*time-invariant system*). Sebuah *frame* ucapan x[n] didapat dari gelombang penuh *signal* ucapan s[n] dengan dikalikan dengan fungsi "jendela" (*window*) w[n] dalam domain waktu [6]:

$$\mathbf{x[n] = w[n]s[n]} \tag{3.2}$$

Ada tiga jenis window yang umum digunakan: *rectangular window, hanning window,* dan *hamming window* yang dinyatakan sebagai berikut:

$$\begin{aligned}
\text{rectangular } w[n] &= \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \\
\text{hanning } w[n] &= \begin{cases} 0.5 - 0.5cos(\frac{2\pi n}{L}) & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \\
\text{hamming } w[n] &= \begin{cases} 0.54 - 0.46cos(\frac{2\pi n}{L}) & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{3.3}$$

Gambar 3.6 di bawah ini adalah pemakaian ketiga window di atas pada gelombang sinusoidal:



(a) Rectangular window

(b) Hanning window

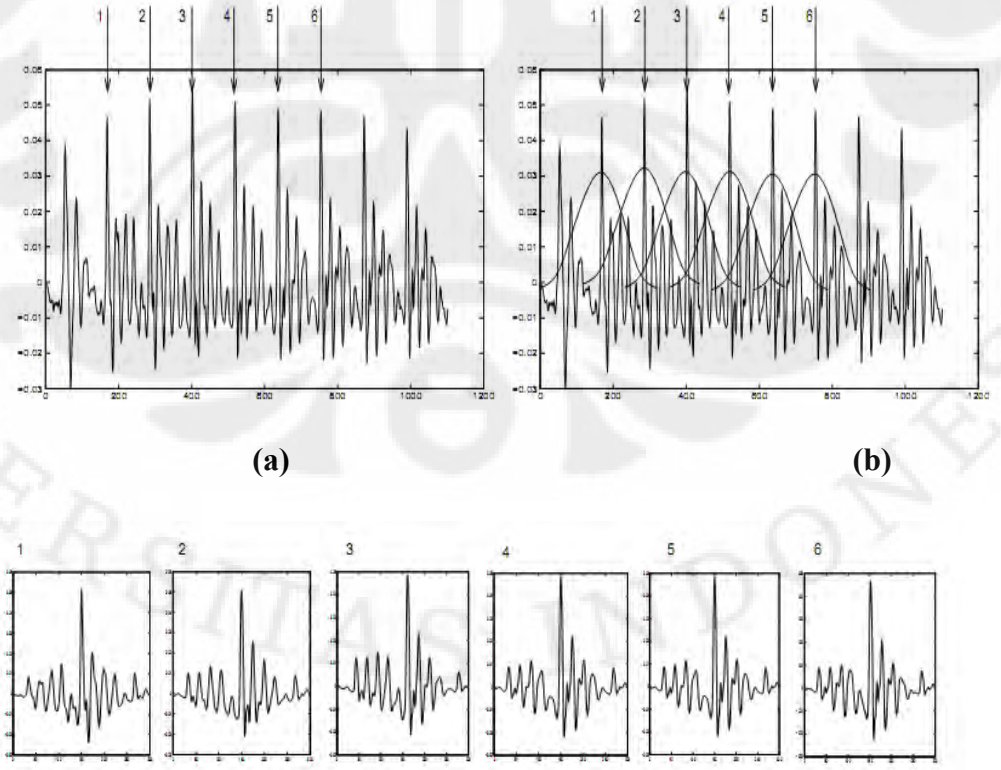
(c) Hamming window

Gambar 3.6. Efek dari windowing pada domain waktu [6]

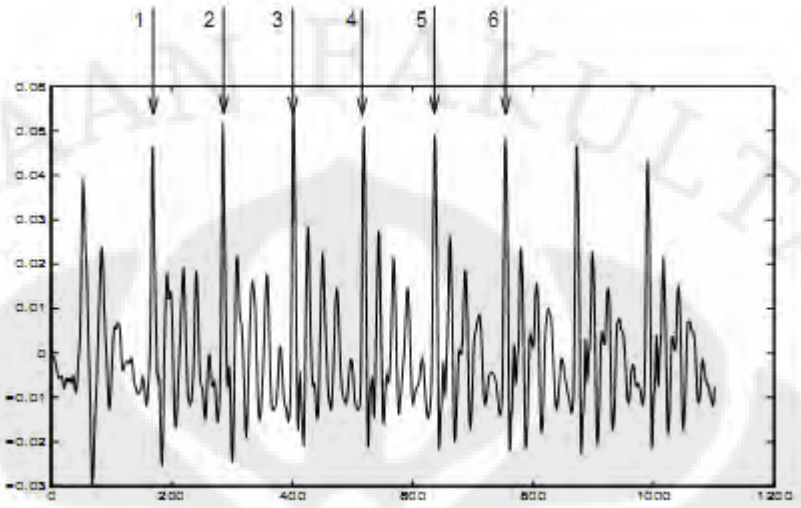
Rectangular window memotong gelombang sinusoid pada ujung *signal*. Sedang *hanning window* dan *hamming window* memiliki kemiripan, yaitu membentuk *envelope* pada *signal*.

TD-PSOLA atau *Time Domain* PSOLA adalah teknik PSOLA yang paling popular. Teknik ini bekerja secara *pitch-synchronously*, yang artinya tiap perioda *pitch* akan dianalisis satu *frame*. Syarat yang dibutuhkan untuk ini adalah kemampuan untuk mengidentifikasi permulaan perioda (*epochs*) pada *signal speech*, dan dengan PSOLA adalah vital untuk dapat melakukannya dengan keakuratan yang tinggi.

Signal ucapan dibagi-bagi menjadi *frame-frame* dengan menggunakan *Hanning Window* yang dikenakan pada perioda *pitch* sebelum dan sesudah *epoch* seperti ditunjukkan pada Gambar (11a). *Frame* hasil *windowing* ini kemudian dapat direkombinasikan dengan meletakkan *epoch frame* tersebut pada posisi yang sama seperti sebelumnya kemudian menambahkan bagian *window* yang bertumpang-tindih dan menambahkannya (oleh sebab itu dinamakan *overlap and add*). Setelah proses ini dilakukan, akan dihasilkan gelombang ucapan yang hampir mirip dengan gelombang ucapan yang asli.



(c)



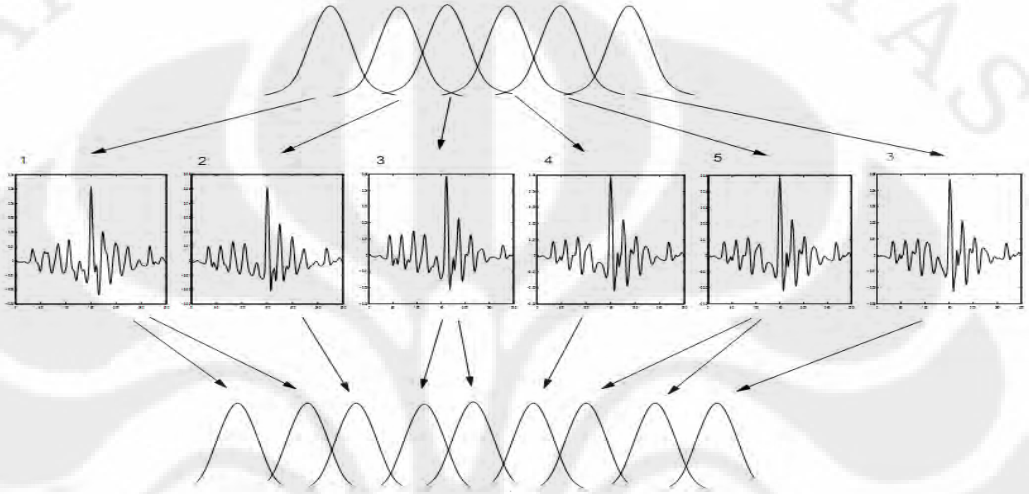
(d)

Gambar 3.7. (a). Suatu bagian dari gelombang ucapan dengan posisi *epoch* yang ditandai dengan gambar anak panah.
(b). Untuk setiap *epoch*, sebuah *frame* dibentuk pada tengah-tengah *epoch* dengan menggunakan *Hanning Window*. Perlu diketahui bahwa *window* tersebut terdapat bagian yang saling menindih (*overlap*).
(c). Kumpulan dari *frame-frame* yang telah dibentuk oleh *Hanning Window*.
(d). Gelombang hasil sintesis ulang dengan menindih dan menambah (*overlap and add*) *frame-frame* yang telah terbentuk di gambar bagian (c) [6].

Meskipun TD-PSOLA banyak digunakan dan sangat efisien, metode ini dapat menghasilkan kecacatan suara pada titik sambungan (*concatenation point*), karena tidak ada cara untuk membuatnya menjadi mulus pada transisi tersebut mengingat kecacatan tersebut terjadi secara mendadak.

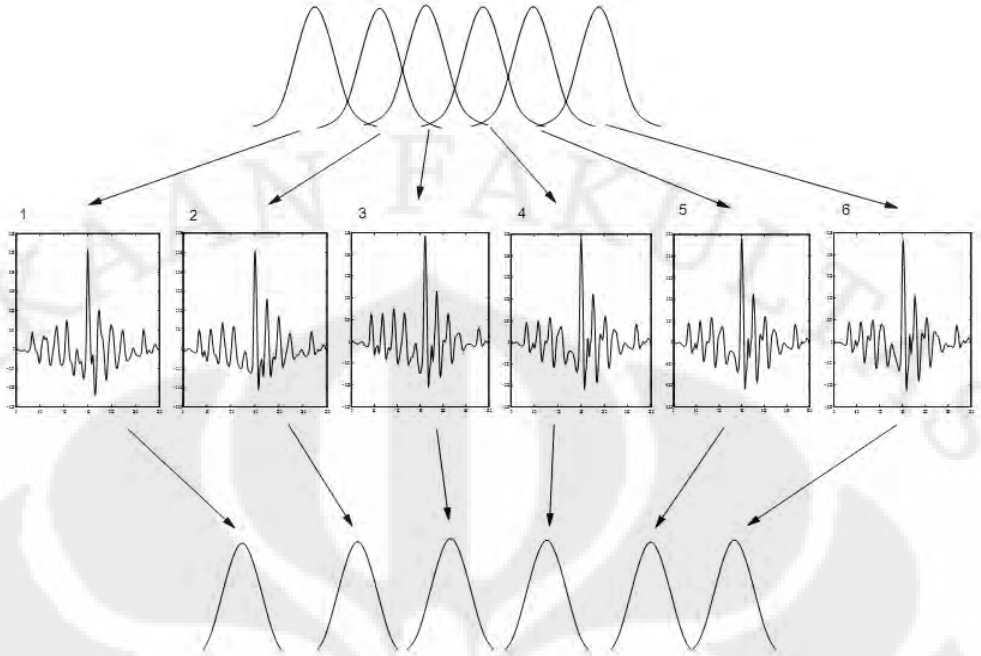
Memanipulasi durasi (*timing*) dalam penggunaan teknik PSOLA ini dilakukan dengan cara menambahkan atau mengeliminasi *frame-frame signal* ucapan tersebut. Misalnya pada Gambar 3.7 di atas, terdapat sebuah fonem ber-

pitch 100 Hz yang dilakukan penjendelaan dengan 6 buah *frame* dan jarak antara *frame* yang pertama dengan *frame* yang terakhir terpisah selama 40 ms. Seseorang dapat menghasilkan segmen suara dengan *pitch* yang sama (100 Hz), namun dengan durasi yang berbeda. Bila seseorang ingin menambah durasi, perlu dilakukan penambahan *frame*. Sedang bila ingin mengurangi durasi, kurangi *frame*. Proses ini dapat dilihat pada Gambar 3.8 di bawah ini.

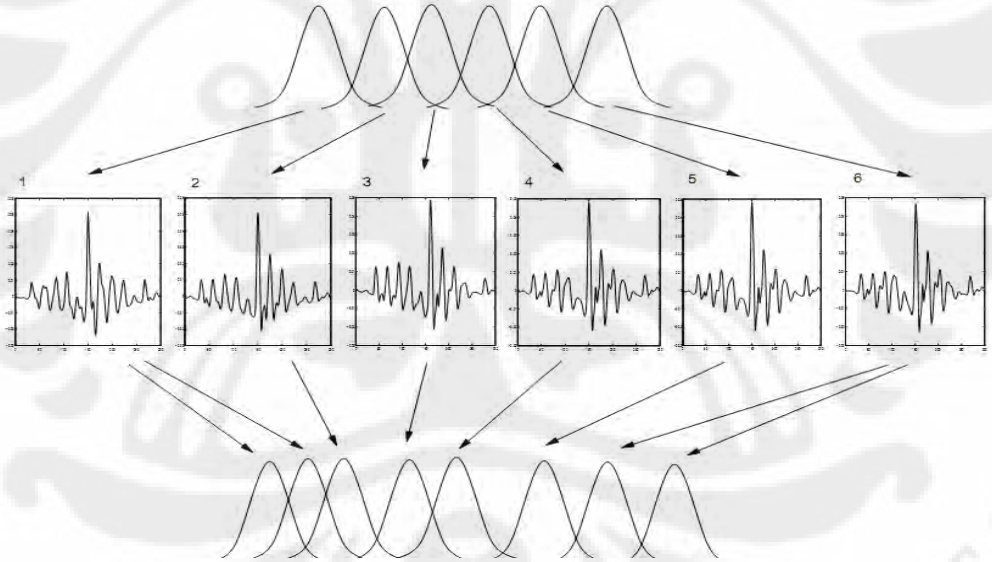


Gambar 3.8. *Timing manipulation* dengan menggunakan teknik PSOLA [6].

Untuk memanipulasi *pitch*, hal ini dilakukan dengan cara menggeser *frame-frame* tersebut sehingga didapatkan *pitch* yang diinginkan. Misalnya, masih pada contoh yang sama, dengan 6 buah *frame* yang besar *pitch* dari fonem adalah 100 Hz. Karena *pitch* sebesar 100 Hz, maka jarak antara satu *frame* dengan *frame* lain adalah 10 ms. Dengan demikian, durasi total segmen kata tersebut yang terdiri dari 6 buah *frame* adalah 40 ms. Bila kita ingin mengubah *pitch* menjadi 150 Hz, maka jarak antara satu *frame* dengan *frame* lain haruslah (1/150) = 0.0066 ms = 6.6 ms. Bila pada awalnya kita memiliki durasi speech 40 ms, maka setelah mengalami perubahan *pitch*, durasi yang baru adalah (6-1)*6,6 ms = 33 ms. Karena jarak antar-*frame* semakin rapat, maka durasi menjadi memendek. Jadi, memanipulasi *pitch* secara tidak langsung juga memanipulasi durasi. Untuk dapat menghasilkan durasi yang sama seperti sebelumnya, maka kita perlu menduplikasi 1 *frame* sehingga (7-1)*6,6 ms = 39,6 ms. Proses *pitch* manipulation dapat dilihat pada Gambar 3.9 di bawah ini.



Gambar 3.9. Pitch manipulation dengan menggunakan teknik PSOLA [6].



Gambar 3.10. Pitch & timing manipulation dengan menggunakan teknik PSOLA [6].

Dalam persiapan pembuatan *database*, ada beberapa tahapan yang harus dilakukan sebelum *synthesizer* dapat menghasilkan kata pertamanya. Tahapan pertama adalah dengan memilih *diphone*, *half-syllables*, dan *triphones* sebagai satuan ucapan (*unit speech*) mengingat mereka terlibat dalam hampir semua

transisi dan koartilulasi seraya agar memori yang digunakan untuk *database* tidak menjadi terlalu besar.

3.2.3. Unit Selection Synthesis

Sekarang ini, *Unit Selection Synthesis* merupakan teknik sintesis yang dominan digunakan dalam TTS. Teknik ini merupakan pengembangan dari *concatenative synthesis*. Teknik ini dikembangkan karena berhubungan dengan permasalahan bagaimana mengatur sejumlah besar unit, mengembangkan prosodi selain menggunakan F0 dan *timing control*, dan bagaimana mengurangi distorsi yang diakibatkan selama pemrosesan *signal*.

Ide dari teknik ini adalah untuk setiap tipe linguistik, terdapat sejumlah unit yang memiliki variasi dalam prosodi dan karakteristik lainnya. Selama proses sintesis, suatu algoritma memilih salah satu unit dari beberapa pilihan yang memungkinkan sebagai usaha dalam menemukan barisan unit yang memenuhi spesifikasi yang dikehendaki.

Dengan perealisasian bahwa untuk tiap *diphone* (bila *diphone* merupakan unit yang dipilih) memiliki satu contoh kata (dengan kata lain memiliki satu unit) akan membatasi kualitas dari sintesis. Jadi, untuk menanggulangi hal ini adalah dengan menyimpan lebih dari satu unit untuk tiap *diphone*. Caranya adalah dengan memperhitungkan fitur lain selain *pitch* dan *timing* (misalnya penekanan atau *stress* dan *phrasing* atau pengelompokan) dan membuat satu unit untuk setiap fitur baru. Jadi sebagai contoh, untuk setiap *diphone*, kita versi yang dimiliki dapat berupa *stressed* dan *unstressed* dan versi *phrase final* dan *non-phrase final*. Sehingga, selain tipe spesifikasi yang digunakan dalam *concatenative* sistem seperti [6]:

$$s_t = \begin{bmatrix} \text{STATE 1} & \begin{bmatrix} \text{PHONEME} & n \\ \text{F0} & 121 \\ \text{DURATION} & 50 \end{bmatrix} \\ \text{STATE 2} & \begin{bmatrix} \text{PHONEME} & t \\ \text{F0} & 123 \\ \text{DURATION} & 70 \end{bmatrix} \end{bmatrix}$$

ditambahkan fitur linguistik tambahan seperti *stress*, *phrasing*, dan sebagainya, sehingga:

$$s_t = \begin{bmatrix} & \begin{bmatrix} \text{PHONEME} & n \\ \text{F0} & 121 \\ \text{DURATION} & 50 \\ \text{STRESS} & true \\ \text{PHARSE FINAL} & false \end{bmatrix} \\ \text{STATE 1} & \\ \text{STATE 2} & \begin{bmatrix} \text{PHONEME} & t \\ \text{F0} & 123 \\ \text{DURATION} & 70 \\ \text{STRESS} & true \\ \text{PHARSE FINAL} & false \end{bmatrix} \end{bmatrix}$$

Salah satu cara untuk mengimplementasikan hal ini adalah dengan mendisain dan merekam data-data yang kita perlukan. Jadi, selain merekam dan menganalisis satu versi untuk setiap *diphone* (bila *database* yang digunakan adalah berupa *diphone*), proses perekaman dan analisis itu juga diterapkan pada tiap kombinasi dari fitur tertentu.

Semakin banyak fitur yang digunakan, maka akan semakin sulit sistem ini digunakan. Hal ini dikarenakan data yang dikumpulkan harus lebih banyak. Prinsipnya, dalam *unit selection synthesis*, dibangun suatu *database* kemudian melakukan suatu analisis sehingga seluruh *database* dapat digunakan sebagai unit dalam sintesis.

Unit Selection dimungkinkan karena tersedianya *database* yang jauh lebih banyak dibandingkan dengan teknik generasi kedua atau *concatenative system*. Dan faktanya adalah percuma memiliki sistem selection yang baik bila pilihan unit adalah terbatas. Dengan *database* yang besar, akan sering ditemukan bahwa bagian *speech* yang panjang dan berderet teratur akan dipilih, dan ini merupakan faktor utama yang bertanggung jawab dalam menghasilkan ucapan yang sangat berkualitas tinggi. Sering terjadi dalam *unit selection synthesis*, tidak ada modifikasi *signal processing* yang dilakukan, dan pendekatan ini disebut sebagai ***pure unit selection***. Sudut pandang lain dalam *unit selection synthesis* adalah ***resequencing algorithm*** yang melakukan pembagian dalam *speech* kemudian mengaturnya kembali. Cara pandang *unit selection* dengan cara ini dikenal

sebagai prinsip modifikasi terkecil (*principle of least modification*). Prinsip ini menjelaskan bahwa kealamian dari original *database* sudah tentu sempurna, dan modifikasi apapun yang kita lakukan, baik itu *trimming, joining*, atau menggunakan *signal processing* akan menghasilkan risiko kerusakan pada data suara yang asli. Sehingga tugas kita adalah menghasilkan spesifikasi yang diinginkan dengan cara mengatur kembali data *original* dalam cara yang sedikit mungkin sampai menghasilkan kualitas suara yang mirip dengan data yang asli.

3.2.3.1. The Hunt and Black Algorithm

Salah satu cara untuk mengatasi permasalahan modifikasi terkecil tadi adalah dengan menggunakan algoritma yang telah dikembangkan oleh Andrew Hunt dan Alan W. Black.

Misalnya suatu spesifikasi yang diinginkan adalah berupa kumpulan *diphone* $S=<s_1,s_2,\ldots,s_T>$ dimana tiap *diphone* digambarkan oleh struktur fiturnya. *Database*-nya merupakan kumpulan dari unit *diphone* $\mathbf{U} = \{U_1, U_2,\ldots,U_M\}$ yang mana tiap unit juga digambarkan oleh suatu struktur fitur. Sistem fitur adalah kumpulan dari fitur-fitur dan nilai-nilai yang digunakan untuk menggambarkan baik spesifikasi maupun unit, dan sistem fitur ini dipilih oleh pembangun sistem (*system builder*) untuk memenuhi sejumlah permintaan. Tujuan dari algoritma unit selection adalah untuk menemukan barisan unit U yang terbaik dari *database* **U** yang memenuhi spesifikasi S.

Dalam kerangka kerja Hunt and Black, *unit selection* didefinisikan sebagai sebuah pencarian melalui semua unit yang memungkinkan untuk menemukan kemungkinan barisan unit yang terbaik. Kata "terbaik" di sini didefinisikan sebagai yang paling menghasilkan biaya termurah dalam perhitungan dua komponen local, yaitu ***Target cost*** $T(u_t,s_t)$ dan ***Join Cost*** $J(u_t,u_{t+1})$. *Target cost* menghitung biaya atau jarak antara spesifikasi s_t dan sebuah unit di dalam *database* u_t. Sedang *join cost* merupakan suatu ukuran seberapa baik dua buah unit terjalin (semakin rendah nilainya, semakin baik jalinan tersebut). Untuk sebuah kalimat, biaya total merupakan gabungan dari *target cost* dan *join cost* [6]:

$$C(U,S) = \sum_{t=1}^{T} T(u_t,s_t) + \sum_{t=1}^{T-1} J(u_t,u_{t+1}) \qquad \textbf{(3.4)}$$

Dan tujuan dari pencarian dalam algoritma Hunt and black ini adalah mencari suatu barisan unit yang meminimalkan biaya itu:

$$\hat{U} = \underset{u}{argmin} \left\{ \sum_{t=1}^{T} T(u_t, s_t) + \sum_{t=1}^{T-1} J(u_t, u_{t+1}) \right\} \quad \textbf{(3.5)}$$

Parameter yang dipakai untuk perhitungan biaya-biaya tersebut (*target sub-cost*) bisa bermacam-macam, seperti *pitch*, durasi, MFCC, daya *signal*, dan property dari *signal* ucapan lainnya. Di dalam skripsi ini, parameter yang digunakan untuk perhitungan *target cost* adalah berupa *pitch* untuk setiap suku kata. Sedang parameter yang digunakan untuk melakukan perhitungan *join cost* adalah *pitch* dan MFCC.

Perhitungan yang sebenarnya dari *target cost* maupun *join cost* sebenarnya memiliki suatu nilai terbobot (*weighted value*). Sehingga rumus yang sebenarnya dari *target cost* dan *join cost* adalah

$$C^t(t_i, u_i) = \sum_{j=1}^{p} w_j^t C_j^t(t_i, u_i) \quad \textbf{(3.6)}$$

$$C^c(u_{i-1}, u_i) = \sum_{j=1}^{q} w_j^c C_j^c(u_{i-1}, u_i) \quad \textbf{(3.7)}$$

Parameter p dan q dari persamaan 3.6 dan 3.7 menyatakan *target sub-cost*. Parameter t menyatakan spesifikasi dari target yagn diinginkan. Sedang parameter u menyatakan spesifikasi dari unit yang terdapat di dalam *database*. Parameter w adalah nilai terbobot itu. Untuk menentukan nilai dari w tersebut perlu dilakukan *training* data. Metode yang dapat digunakan dapat berupa *Hidden Markov Model* (HMM), *Kullback-Leibler Divergence* (KLD).

Penjelasan MFCC akan dimulai dari apa itu *Short Term Speech Analysis*. Penjelasannya adalah sebagai berikut.

Hal yang paling sering dilakukan sebagai titik mula dalam melakukan analisis ucapan adalah dengan menemukan *magnitude* dari *spectrum signal*

ucapan (*speech signal*). *Signal* ucapan merupakan salah satu contoh dari *signal* yang tidak stasioner (*non-stationary signal*). Sedangkan bila ingin melakukan analisis terhadap *signal* jenis ini, khususnya bila menganalisisnya dengan menggunakan analisis Fourier, maka mau tidak mau seseorang harus melakukan sesuatu terhadap *signal* yang tidak stasioner ini sehingga dapat dilakukan analisis dengan menggunakan analisis Fourier.

Presuposisi dari analisis Fourier adalah bahwa *signal* yang dianalisis bersifat stasioner. *Signal* dikatakan bersifat non-stasioner bila memiliki properti intrinsik yang bervariasi terhadap waktu. Sehingga, agar seseorang dapat menganalisis *signal* yang tidak stasioner, maka salah satu caranya adalah dengan menganalisisnya pada selang-selang waktu yang pendek (*short time*), dengan mengasumsikan bahwa *signal* ucapan tersebut adalah stasioner pada selang waktu yang cukup pendek tersebut (*signal* ucapan sangat tidak stasioner, namun pada selang waktu sekitar 20 ms, property dari *signal* tidak berubah secara signifikan [11]). Alasan lainnya adalah bahwa seringkali yang menjadi masalah fundamental adalah diperlukannya informasi mengenai interval waktu tertentu ketika suatu frekuensi tertentu terjadi. Oleh karena analisis yang dilakukan dengan cara seperti ini, maka analisis Fourier demikian disebut sebagai *Short-Time Fourier Analysis* (STFT). STFT memungkinkan untuk dilakukan analisis *time-frequency*, atau dengan kata lain, *signal* yang dianalisis dapat direpresentasikan dalam bidang *time-frequency*.

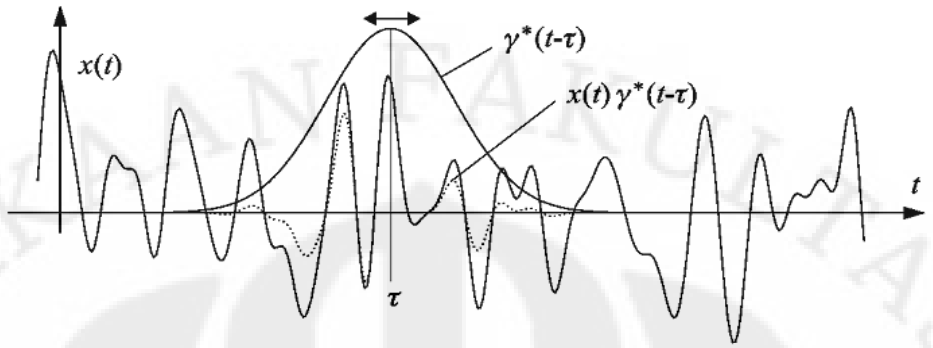
Misalnya *signal* yang ingin dianalisis adalah x(t), dan $\gamma^*(t-\tau)$ adalah suatu fungsi jendela (*window function*). Dengan demikian, *signal* $x_\tau(t)$ yang adalah *signal* yang telah dimodifikasi didefinisikan sebagai [11]:

$$X_\tau(t) = x(t)\,\gamma^*(t-\tau) \qquad \textbf{(3.6)}$$

dimana τ adalah suatu parameter *delay*.

Representasi Fourier untuk *signal* ucapan yang telah dimodifikasi tersebut adalah [11]:

$$\mathcal{F}_x^{\gamma}(\tau,\omega) = \int_{-\infty}^{\infty} x(t)\;\gamma^*(t-\tau)\;e^{-j\omega t}\;dt. \qquad \textbf{(3.7)}$$

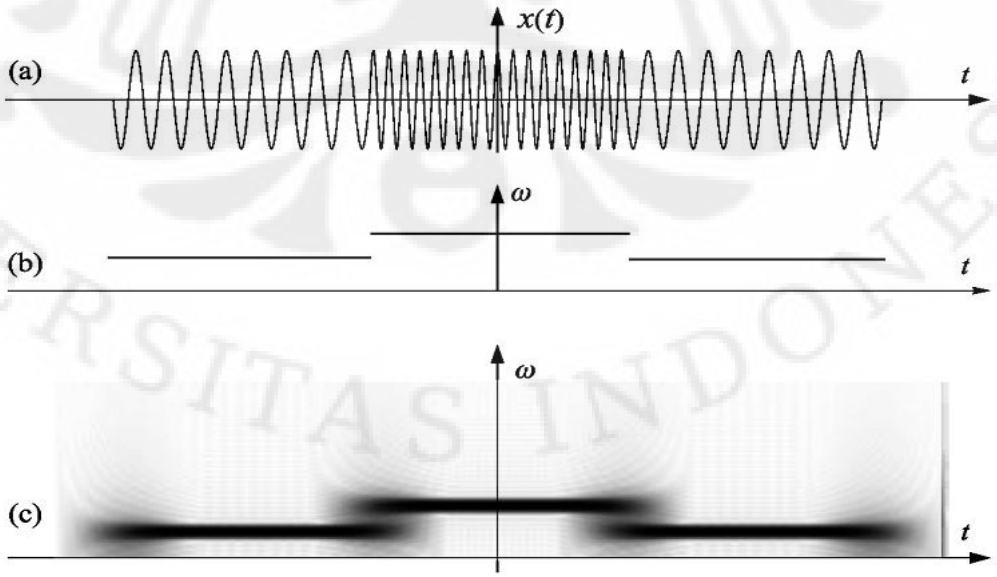


Gambar 3.11 . Short-Time Fourier Transform [11].

Spectrogram merupakan salah satu contoh aplikasi dari STFT. *Spectrogram* merupakan plot 3 dimensi dari *signal* yang menampilkan properti dari *signal* berupa waktu (di sumbu horizontal), frekuensi (di sumbu vertikal), dan energi dari *signal* (tingkat kegelapan). Secara matematis, *spectrogram* ini didefinisikan sebagai kuadrat dari magnitude STFT *signal* yang bersangkutan [11]:

$$S_x(\tau,\omega) = |\ \mathcal{F}_x^{\gamma}(\tau,\omega)|^2 = \left|\ \int_{-\infty}^{\infty} x(t)\ \gamma^*(t-\tau)\ e^{-j\omega t}\ dt\ \right|^2 \quad \textbf{(3.8)}$$

Gambar dari *spectrogram* dapat dilihat pada Gambar 3.12 berikut. :



Gambar 3.12 . a. *Signal* uji coba.

b. Representasi time-frequency ideal dari *signal* uji coba

c. *Spectrogram* [11]

Cepstrum didefinisikan sebagai inverse DFT dari logaritma dari magnitude DFT suatu *signal*:

$$c[n] = \mathcal{F}^{-1}\{\ log|\ \mathcal{F}\ \{x[n]\}\ |\ \} \tag{3.9}$$

Dengan demikian, *cepstrum* dari *signal* x[n] yang telah mengalami penjendelaan adalah

$$c[n] = \sum_{n=0}^{N-1} log\left(\left| \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \right| \right) e^{j\frac{2\pi}{N}kn} \tag{3.10}$$

MFCC memiliki kesamaan dengan *cepstrum*, hanya bedanya pada MFCC, *magnitude* spektrumnya berada pada skala mel (diambil dari kata melody untuk mengindikasikan bahwa skala ini berdasarkan pada perbandingan *pitch* [12]). Skala mel didefinisikan sebagai:

$$m = 2595 \log_{10}\left(\frac{f}{700} + 1\right) = 1127 \log_e\left(\frac{f}{700} + 1\right) \tag{3.11}$$

Spektrum yang direpresentasikan oleh *mel-cepstral coefficients* memiliki resolusi frekuensi yang mirip dengan telinga manusia. Akibatnya, *mel-cepstral coefficients* berguna untuk sintesis ucapan (*speech synthesis*) dan pengenalan ucapan (*speech recognition*). MFCC memiliki sifat yang relatif sensitif terhadap penambahan distorsi ketidakcocokan [13]. Hanya dengan menambahakan white

noise kepada *signal* ucapan akan mempengaruhi spectrum daya *signal* pada semua frekuensi. Namun, Dampak tersebut tidak sama untuk semua range frekuensi, dengan kata lain, nilai SNR adalah berbeda untuk semua frekuensi.

Langkah-langkah untuk menentukan MFCC adalah sebagai berikut:

1. Lakukan transformasi Fourier pada *signal* yang telah mengalami penjendelaan.
2. Hitung magnitude dari hasil pertama di atas lalu ubah ke dalam skala mel.
3. Ambil nilai logaritma dari magnitude berskala mel tersebut.
4. Lakukan discrete cosine transform terhadap daftar magnitude *signal* berskala mel itu dengan menganggap bahwa daftar daya itu adalah *signal*.
5. MFCC adalah amplitude dari spectrum yang dihasilkan.

Discrete Cosine Transform didefinisikan sebagai:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right) \qquad \textbf{(3.12)}$$

BAB IV

DISAIN SISTEM *TEXT-TO-SPEECH* BERBASISKAN *UNIT SELECTION SYNTHESIS* UNTUK BAHASA INDONESIA

4.1. Proses Pembuatan Sistem TTS

Sistem TTS yang dibuat memiliki 12 modul yang diprogram dengan menggunakan software Borland Delphi 7 sebagai software utamanya dan MySQLFront sebagai system *database*-nya. Kedua belas modul itu antara lain:

1. Modul TTS.

   Modul ini merupakan modul utama yang disimpan dengan nama TTS.pas. Di dalam modul ini sudah tergabung 2 modul, yaitu modul Prosody Predicting (berupa prosedur Prosody Predicting) dan modul Unit Selection Synthesis (berupa prosedur TargetCost, JoinCost, dan TotalCost).

2. Modul SentenceBreaking

   Modul ini terdapat di dalam unit yang diberi nama SentenceBreaking.pas. Modul ini berfungsi untuk melakukan *parsing* terhadap *input* teks sehingga dapat diproses untuk memperoleh kumpulan kalimat-kalimat yang utuh. Kalimat utuh yang dimaksudkan di sini adalah kalimat yang diakhiri dengan tanda baca seperti tanda titik (.), tanda seru (!), dan tanda Tanya (?). Kalimat-kalimat tersebut kemudian dimasukkan ke dalam MemoSentence (yang berjudul Sentences from *Input* Text).

3. Modul WordParsing

   Modul ini terdapat di dalam unit yang diberi nama WordParsing.pas. Modul ini berfungsi untuk memecah teks *input*an menjadi kumpulan kata-kata. Kata-kata yang telah dibentuk diberi tanda # di akhir kata itu bila kata tersebut tidak terdapat tanda baca apapun. Bila di akhir suatu kata terdapat tanda baca seperti titik (.), koma (,), tanda seru (!), atau tanda Tanya (?), maka tidak diberi tambahan tanda #. Misalnya terdapat kalimat "Budi pergi ke pasar.". Maka kata-kata yang dibentuk adalah "Budi#", "pergi#", "ke#', dan "pasar.". Kata-kata tersebut kemudian dimasukkan ke dalam ListBox2 (yang berjudul Word List from Input Text).

4. Modul Checking

Modul ini terdapat di dalam unit yang diberi nama Checking.pas. Modul ini berfungsi untuk melakukan kegiatan yang berupa pengecekan. Pengecekan itu berupa pengecekan terhadap suatu kata apakah kata tersebut berformat angka, tanggal, atau waktu, serta menentukan jenis suku kata apakah suku kata itu termasuk tipe 0 (bila merupakan bukan suku kata terakhir), tipe 1 (bila di akhir suku kata itu terdapat tanda titik), tipe 2 (bila di akhir suku kata itu terdapat tanda koma), tipe 3 (bila di akhir suku kata itu terdapat tanda seru), tipe 4 (bila di akhir suku kata itu terdapat tanda tanya), dan tipe 5 (bila di akhir kata itu terdapat tanda #).

Di dalam modul ini, teks *input* yang dimasukkan di MemoInput (yang judulnya Input Text) akan diproses sehingga menghasilkan kalimat-kalimat utuh. Setelah teks menjadi kalimat utuh, proses berlanjut lagi sehingga diperoleh kata. Kata yang dibentuk dibubuhi di akhir tiap kata dengan tanda # kata yang tidak memiliki tanda baca sama sekali di akhir kata tersebut. Untuk kata yang memiliki tanda baca (titik, koma, seru, dan tanya), tanda # tidak dibubuhkan. Pembubuhan tanda ini berfungsi untuk membantu proses penentuan prosodi nantinya.

5. Modul NumbNormalizing

   Modul ini terdapat di dalam unit yang diberi nama NumbNormalizing.pas. Modul ini berfungsi untuk melakukan normalisasai terhadap *input* kata berupa angka. Angka yang dapat diproses oleh modul ini baru sampai batas juta saja (maksimum 9.999.999) dan dapat juga memroses angka yang ada di belakang koma. Bila *input* kata lebih dari 7 digit, maka akan dilafalkan per-angka.

6. Modul DateNormalizing

   Modul ini terdapat di dalam unit yang diberi nama DateNormalizing.pas. Modul ini berfungsi utnuk melakukan normalisasi terhadap *input* kata yang berupa tanggal. Format tanggal yang dapat diproses oleh modul ini adalah berupa dd-mm-yy atau dd-mm-yyyy atau mm-dd-yy atau mm-dd-yyyy atau yy-mm-dd.

7. Modul TimeNormalizing

Modul ini terdapat di dalam unit yang diberi nama TimeNormalizing.pas. Modul ini berfungsi untuk melakukan normalisasi terhadap *input* kata yang berupa waktu. Format waktu yang dapat diproses oleh modul ini adalah berupa hour-min dan hour-min-sec.

8. Modul SyllableParsing

   Modul ini terdapat di dalam unit yang diberi nama SyllableParsing.pas. Modul ini berfungsi untuk melakukan *parsing* terhadap kata yang masuk ke modul ini.

9. Modul ProsodyPredicting

   Modul ini terdapat di dalam modul utama (modul TTS) berupa suatu prosedur yang diberi nama ProsodyPredicting. Modul ini bertugas untuk menentukan property dari *signal* ucapan *input*an yang masuk ke modul ini (berupa suku kata). Properti *signal* ucapan yang diaplikasikan di dalam modul ini antara lain: *pitch*, durasi, ToBI, dan MFCC.

10. Modul Unit Selection Synthesis

    Modul ini terdapat di dalam modul utama berupa 4 buah prosedur, yaitu prosedur TargetCost, JoinCost, TotalCost, dan SearchingFile. Modul TargetCost, JoinCost, dan TotalCost bertugas untuk menentukan cost dari tiap suku kata. Lalu, setelah cost-cost telah didaftar, oleh modul SearchingFile akan mencari pasangan *file* suku kata-suku kata yang memberntuk kata dengan nilai *total cost* yang paling kecil.

11. Modul PlayingSound

    Modul ini terdapat di dalam unit yang diberi nama PlayingSound.pas. Tugas dari modul ini adalah memainkan *file* suku kata yang telah mengalami proses penyeleksian oleh modul Unit Selection Synthesis. Modul ini baru dapat mengeluarkan suara dari kata yang mengandung suku kata paling banyak 3 buah saja.

12. Modul Tools

    Modul ini terdapat di dalam unit yang diberi nama Tools.pas. Modul ini berfungsi sebagai semacam alat bantu bagi modul-modul lainnya dalam melakukan proses di dalam modul yang bersangkutan, seperti me-reset suatu

array, pengurutan data, pencarian koordinat, mencari jumlah kata atau suku kata atau elemen array, dan parsing kata dari exceptional dictionary.

Untuk membuat *database* ucapan, digunakan software WavePad Sound Editor v 4.27 untuk mengedit suatu rekaman yang akan menjadi *database* ucapan dan Praat v 5.1.0.7 untuk menentukan property dari *signal* ucapan tersebut. *Database* ucapan yang dibuat adalah berupa suku kata yang diperoleh dari hasil pengeditan suatu rekaman ucapan. Rekaman ucapan itu diedit dengan menggunakan software WavePad Sound Editor untuk memperoleh penggalan suku kata-suku kata. Pemenggalan yang dilakukan tidak dilakukan dengan teknik yang khusus, hanya mengandalkan persepsi dari pendengaran saja untuk menentukan suatu suku kata kemudian dilakukan pemenggalan. Suku kata yang diperoleh itu kemudian disimpan dalam format *wav* dengan nama yang sesuai dengan suku kata itu beserta dengan nomor urut bila ada suku kata yang sama.

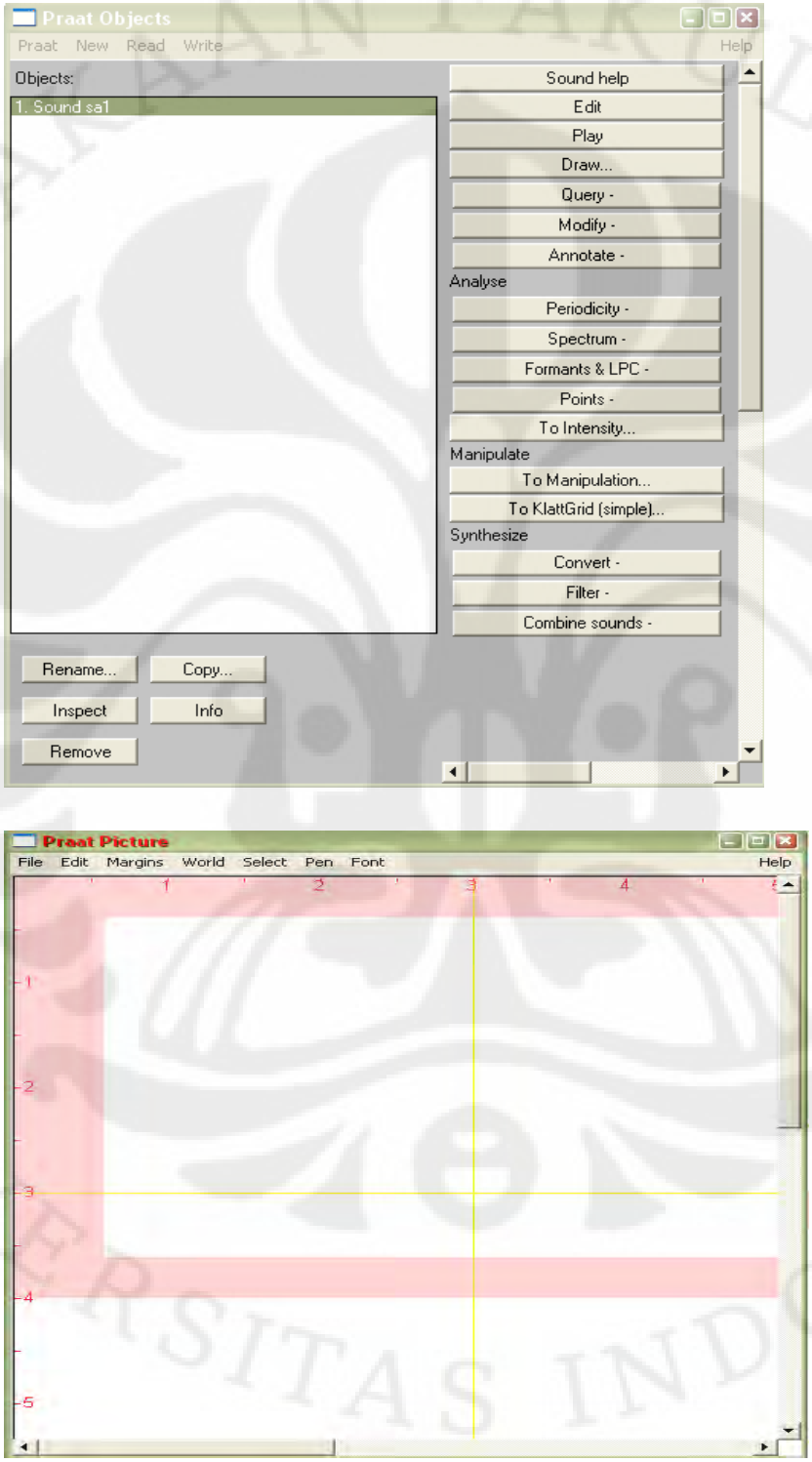
File suku kata itu kemudian diproses lagi dengan menggunakan software Praat untuk ditentukan *pitch*, durasi, MFCC, dan ToBI. Data-data tersebut disimpan informasinya ke dalam MySQL. Data-data ini diperlukan oleh modul NLP untuk menentukan *file* suku kata mana yang akan digunakan setelah diproses oleh modul Unit Selection Synthesis.

Praat merupakan software *speech analyser* yang sangat ampuh dan sering dipakai untuk melakukan analisis ucapan. Terlebih lagi, software ini merupakan software yang gratis dan dapat diunduh dari internet. Dengan software ini, kita dapat menentukan beragam macam property dari suatu *signal* ucapan. Di dalam skripsi ini, property dari *signal* ucapan yang akan dicari nilainya adalah *pitch*, durasi, ToBI, dan MFCC.

Properti dari *signal* suatu suku kata yang berupa *pitch* diperoleh dengan mengambil data dari table mprosody di *database* yang merupakan nilai *pitch* rata-rata dari semua suku kata yang sama lalu ditambah dengan suatu nilai random. Jadi, misalnya ingin ditentukan nilai *pitch* dari suku kata “sa”. Anggap di dalam *database* terdapat 3 buah suku kata “sa” yang masing-masing memiliki *pitch* x,y,z. Nilai *pitch* rata-ratanya adalah (x+y+z)/3. Nilai rata-rata ini yang diambil,

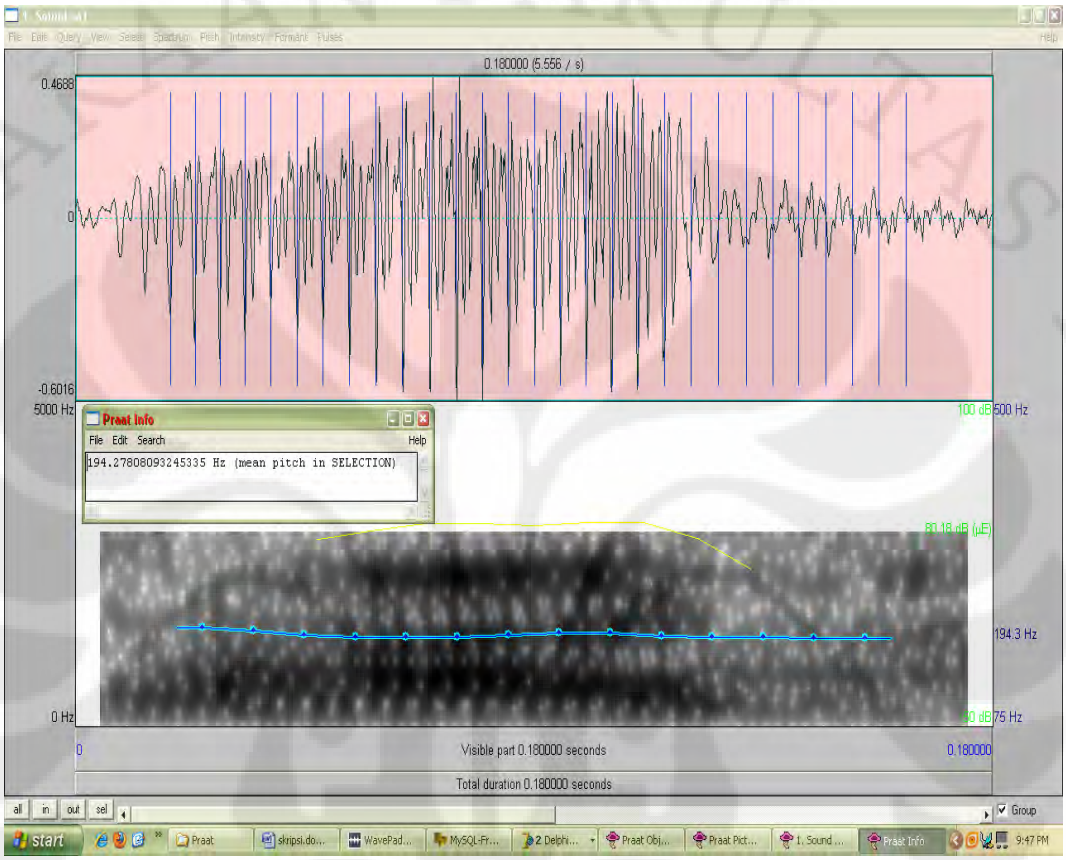
kemudian ditambah dengan suatu nilai random. Dan hasilnya adalah nilai *pitch* akhir untuk suku kata "sa" tadi.

Bentuk tampilan dari Praat dapat dilihat pada Gambar 4.1 di bawah ini:



Gambar 4.1. Tampilan Praat.

Misalnya kita ingin menentukan property dari *signal* suku kata "sa". Tampilan dari *signal* ucapan suku kata ini pada Praat dapat dilihat pada Gambar 4.2 di bawah ini:

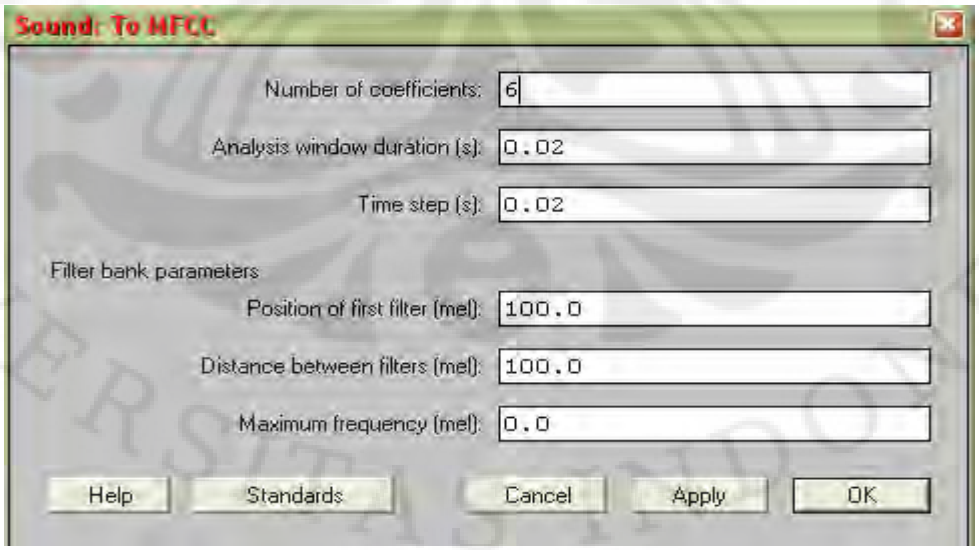


Gambar 4.2. Suku kata "sa" beserta dengan propertinya yang ditampilkan oleh Praat.

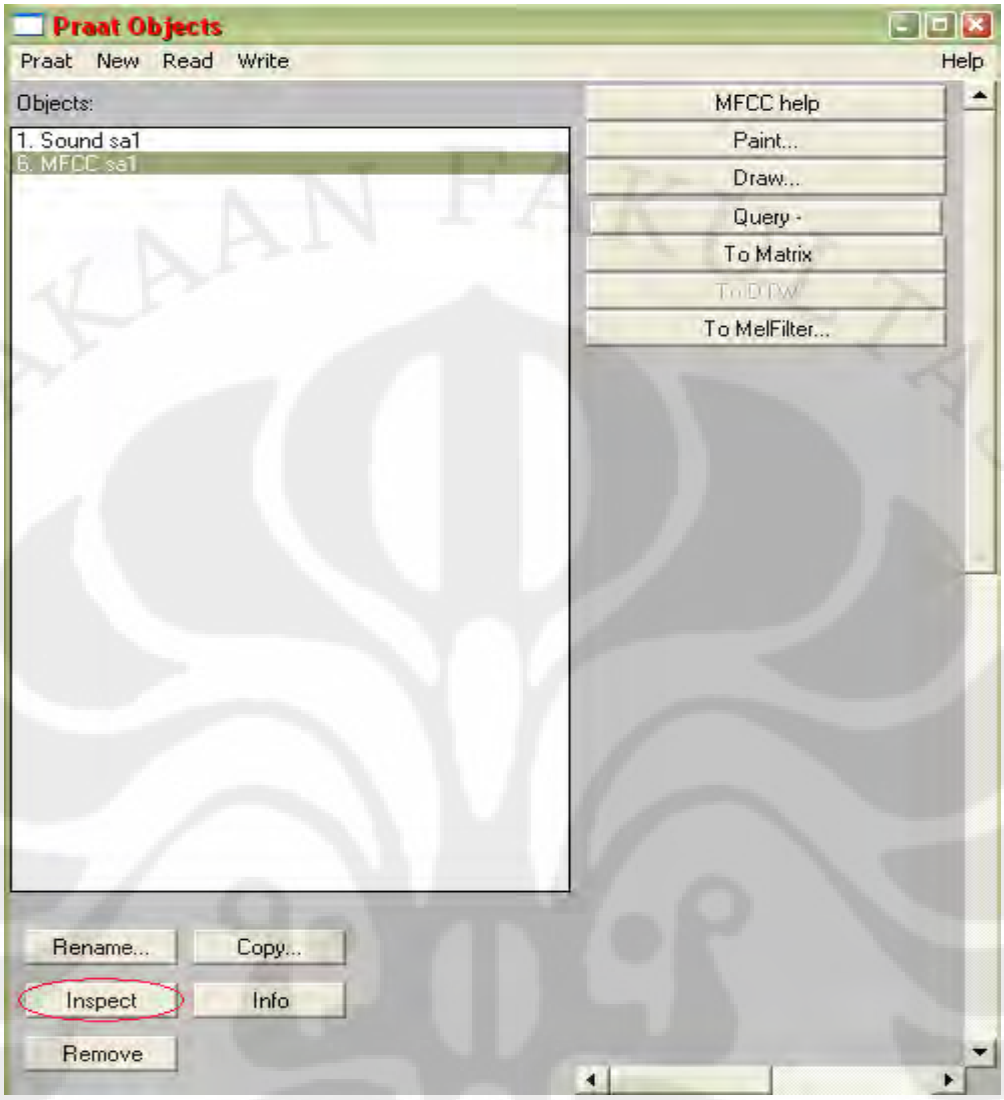
Properti dari *signal* suku kata "sa" yang dapat kita lihat pada Gambar 4.2 di atas antara lain: *signal* ucapan suku kata itu sendiri (layar berwarna pink), *spectrogram* (layar berwarna kehitam-hitaman), *pitch* tracking (garis berwarna biru di atas *spectrogram*), durasi, dan *pitch* rata-rata.

Untuk menentukan nilai MFCC, hal yang perlu dilakukan adalah dengan menekan tombol *Formant*s & LPC untuk *file* suara yang ingin dicari nilai MFCC-nya pada program Praat. Untuk lebih jelasnya dapat dilihat pada Gambar 4.3 di bawah ini:

(a)



(b)



(c)

(d)

(e)

Gambar 4.3. Langkah-langkah mendapatkan nilai MFCC pada Praat. (a). Langkah pertama. (b). Langkah kedua. (c). Langkah ketiga. (d). Langkah keempat. (e). Langkah kelima.

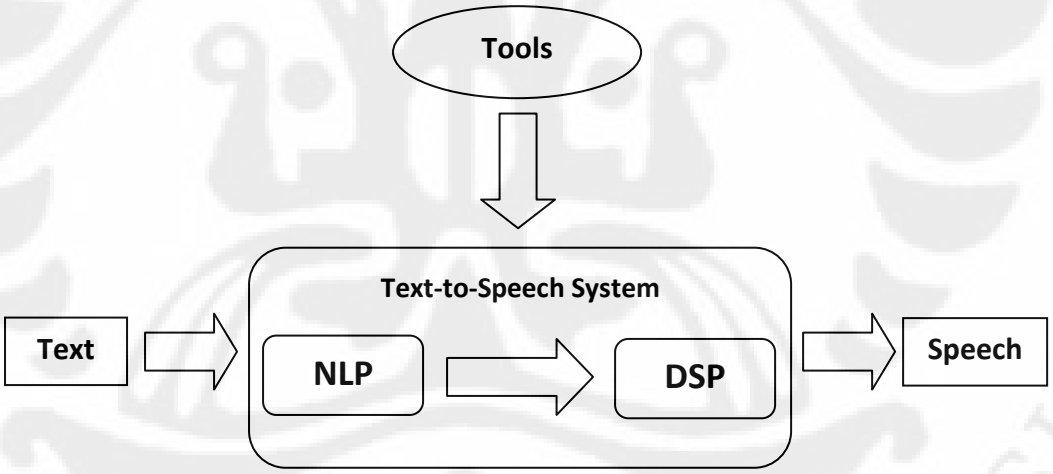
Setelah langkah pertama akan muncul kotak dialog untuk menentukan berapa jumlah koeffisien yang diinginkan, durasi dari window, langkah waktu (time step) antar window, dan parameter dari bank filter. Jumlah dari koeffisien

di-set menjadi 6 buah, durasi serta time step diset menjadi 20 ms, dan parameter lainnya bernilai *default* (lihat Gambar 4.3b).

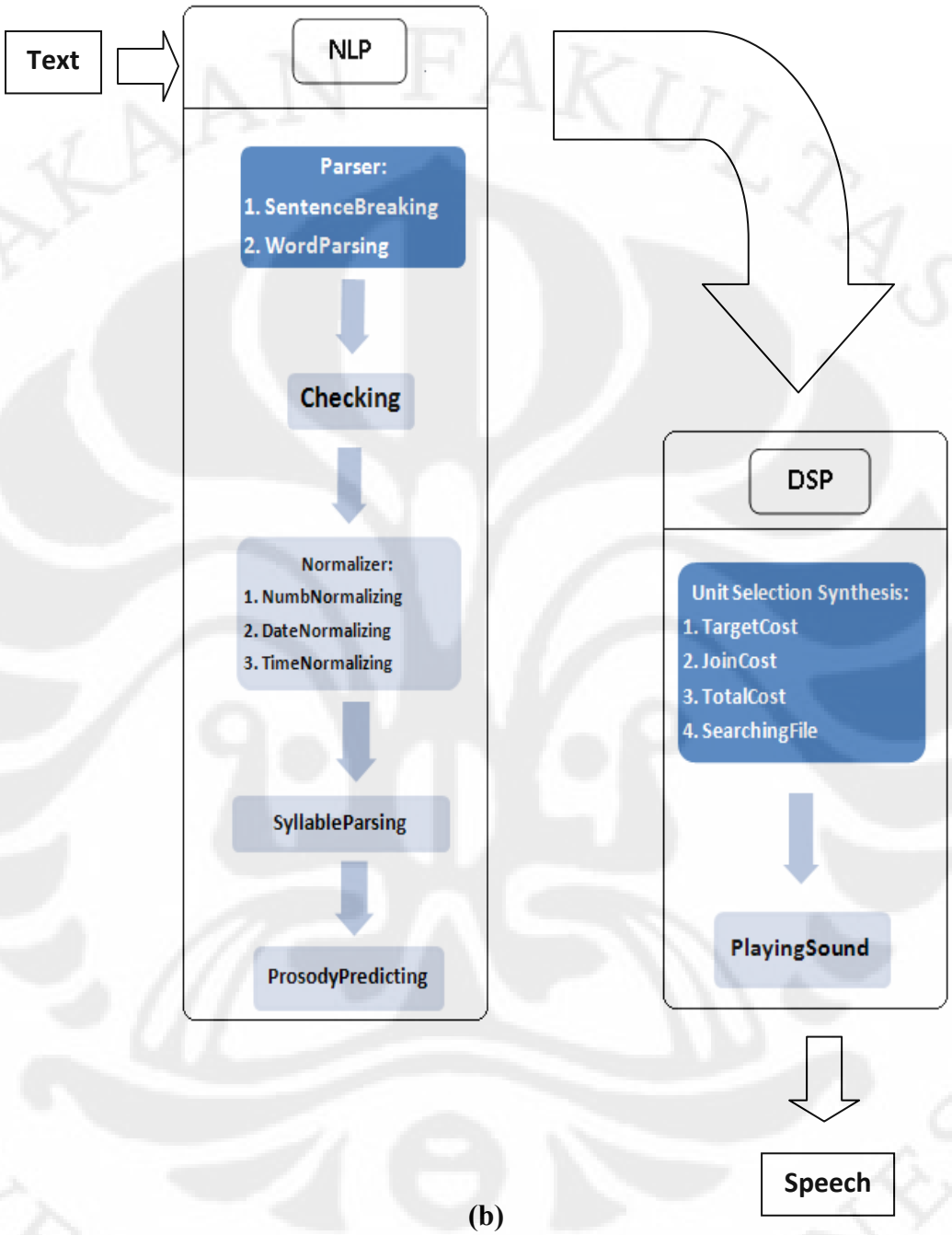
Kemudian klik tombol ok maka akan tampak seperti pada Gambar 4.3c. Kemudian klik tombol Inspect yang berada di kiri bawah sehingga muncul kotak lagi seperti pada Gambar 4.3d. Klik lagi tombol open yang ada di bawah sehingga muncul kotak lagi yang menampilkan nilai dari MFCC seperti pada Gambar 4.3e. Nilai MFCC yang diambil adalah nilai dari koefisien ke-0 atau c0 untuk 6 *frame* yang pertama karena menurut [MFCC], koefisien ke-0 ini dapat diinterpretasikan sebagai tingkat kekerasan (*loudness*) yang dirata-ratakan pada window segitiga. Data-data tersebut kemudian dimasukkan ke dalam *database*. Cara yang sama juga dilakukan untuk suku kata yang lain.

4.2. Arsitektur dan Algoritma dari Sistem TTS yang Dibuat

Arsitektur dari system TTS yang dibuat di skripsi ini, secara garis besar dapat dilihat pada Gambar 4.4 di bawah ini:



(a)



(b)

Gambar 4.4. Arsitektur system TTS yang dibuat. (a). Secara garis besar. (b). Secara lebih detail

Sistem TTS yang dibuat pada skripsi ini memiliki prinsip yang mirip seperti

yang telah dijelaskan sebelumnya di bagian dasar teori. Sistem TTS ini memiliki 2 modul utama, yaitu modul NLP (*Natural Language Processing*) dan modul DSP (*Digital Signal Processing*). Modul NLP disusun atas 5 modul, yaitu

1. Modul Parser

   Modul ini bertugas untuk memecah *input* teks yang masuk agar dapat dipecah menjadi kalimat (tugas ini dikerjakan oleh unit SentenceBreaking) lalu dipecah lagi menjadi kata (tugas ini dikerjakan oleh unit WordParsing).

2. Modul Checking

   Modul ini menerima masukan *input* berupa kata dari modul parser. *Input* tersebut kemudian dilakukan pemeriksaan apakah *input* tersebut merupakan angka, tanggal, waktu, atau berupa kata biasa saja. Informasi berupa status kata tersebut kemudian dikirim ke modul selanjutnya, yaitu modul normalizer.

   Modul ini sebenarnya juga melakukan pemeriksaan apakah kata *input*an merupakan kata yang terdapat di dalam *database* atau tidak. Namun, *command* untuk melakukan tugas ini tidak diletakkan di dalam modul ini. *Command* tersebut diletakkan di dalam modul utama (TTS.pas). Jadi, modul ini dengan kata lain melakukan 4 buah pengecekan: angka, tanggal, waktu, dan exceptional dictionary (exceptional dictionary yang terdapat di dalam skripsi ini berupa *database* yang berisi daftar akronim (dbacronim) dan *database* yang berisi daftar kata-kata asing dan kata pengecualian (dbexcpdict)).

3. Modul Normalizer

   Informasi dari modul checking yang berupa status dari kata apakah kata *input*an berupa angka, tanggal, waktu, atau kata biasa, dipakai oleh modul ini untuk melakukan normalisasi yang sesuai dengan status kata. Bila status kata berupa angka, maka kata tersebut akan dinormalisasi oleh modul NumbNormalizing. Bila statusnya adalah tanggal, maka akan dinormalisasi oleh modul DateNormalizing. Bila status adalah waktu, maka akan dinormalisasi oleh modul TimeNormalizing. Bila statusnya adalah kata biasa saja, maka *input* kata tersebut akan dibiarkan saja.

   Modul ini sebenarnya juga melakukan normalisasi terhadap kata *input*an yang terdapat di dalam *exceptional dictionary*. Namun *command-command* untuk

melakukan tugas ini tidak diletakkan di dalam modul ini, melainkan di dalam modul Tools. Bila keluaran dari modul checking menyatakan bahwa kata *input*an terdapat di dalam *database*, maka normalisasi dilakukan oleh modul Tools (oleh prosedur ExtractWrdFromDict).

4. Modul SyllableParsing

   Modul ini bertugas untuk memecah *input* kata yang telah dinormalisasi menjadi suku kata. Modul ini terletak di dalam modul ProsodyPredicting. Meskipun demikian, penulis menganggap modul ini terpisah dari modul ProsodyPredicting karena tugas yang dikerjakan oleh modul ini memegang peranan yang cukup krusial di dalam system TTS.

5. Modul ProsodyPredicting

   Modul ini bertugas untuk menentukan property dari *signal* suku kata yang masuk. Properti *signal* yang ditentukan adalah berupa *pitch*, durasi, ToBI, dan MFCC.

Keluaran dari modul NLP adalah suku kata berikut dengan informasi prosodinya. Keluaran ini kemudian masuk ke modul DSP untuk diproses lagi sehingga dapat dihasilkan ucapan. Modul DSP ini terdiri dari 2 modul utama, yaitu:

1. Modul Unit Selection Synthesis

   Modul ini merupakan modul penyintesa. Modul ini terdiri dari 4 prosedur yang diletakkan di dalam modul utama. Prosedur TargetCost, JoinCost, TotalCost, dan SearchingFile. Ketiga modul yang pertama bertugas untuk membuat daftar cost yang kemudian hasil *total cost* tersebut disimpan di dalam suatu array 1 dimensi. Oleh prosedur SearchingFile, array 1 dimensi yang berisi daftar *total cost* untuk semua kemungkinan pasangan suku kata tersebut diproses lagi untuk mendapatkan indeks yang mana adalah alamat dari *total cost* yang terendah. Setelah ditemukan indeks tersebut, maka akan dilakukan suatu perhitungan sederhana untuk menentukan *file* suku kata mana saja yang akan dimainkan oleh modul PlayingSound.

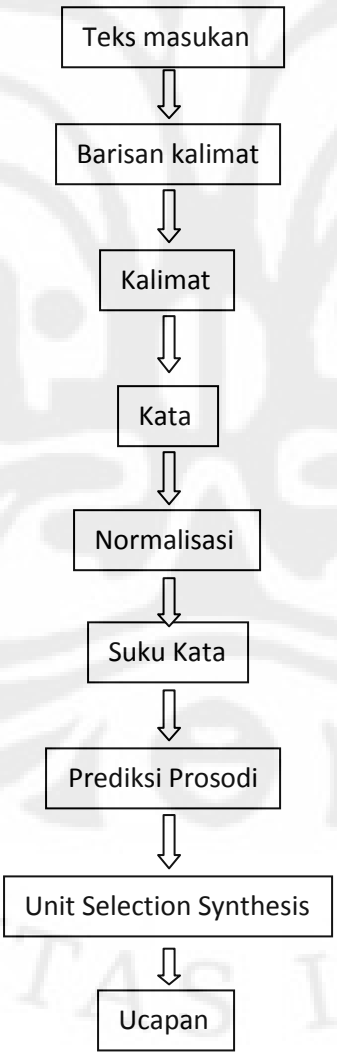
   Prosedur SearchingFile dijalankan di dalam prosedur TotalCost. Penulis menganggap prosedur ini dibedakan meskipun dijalankan di dalam prosedur

TotalCost untuk memberi pengertian bahwa dalam system ini terjadi pencarian *file* karena prosedur pencarian *file* ini terletak di dalam prosedur yang memiliki nama seakan-akan tidak terjadi pencarian *file*.

2. Modul PlayingSound

   Modul ini menerima *input*an dari prosedur SearchingFile berupa *file wav* mana saja yang akan dimainkan. Kumpulan *file wav* suku kata tersebut akan dibuka dengan perintah PlaySound sehingga akan dihasilkan ucapan kata yang terbentuk dari gabungan suku kata-suku kata.

Secara garis besar, cara kerja dari system TTS yang dibuat dapat dilihat pada flowchart pada Gambar 4.5 di bawah ini:



Gambar 4.5. Flowchart sistem TTS yang dibuat.

Yang menjadi masukan ke dalam system ini memang berupa teks yang dapat terdiri dari beberapa kalimat. Namun, yang diproses hingga menghasilkan ucapan adalah tiap kata dari teks tersebut. Jadi, teks yang dimasukkan akan diolah sedemikian rupa sehingga menghasilkan kumpulan kata-kata yang disimpan di dalam ListBox2 (yang berjudul Word List from *Input* Text). Kata-kata tersebut kemudian diproses lagi untuk ditentukan property *signal* yang bersesuaian dengan kata tersebut. Tugas ini dikerjakan oleh modul ProsodyPredicting yang terdapat di dalam modul TTS.

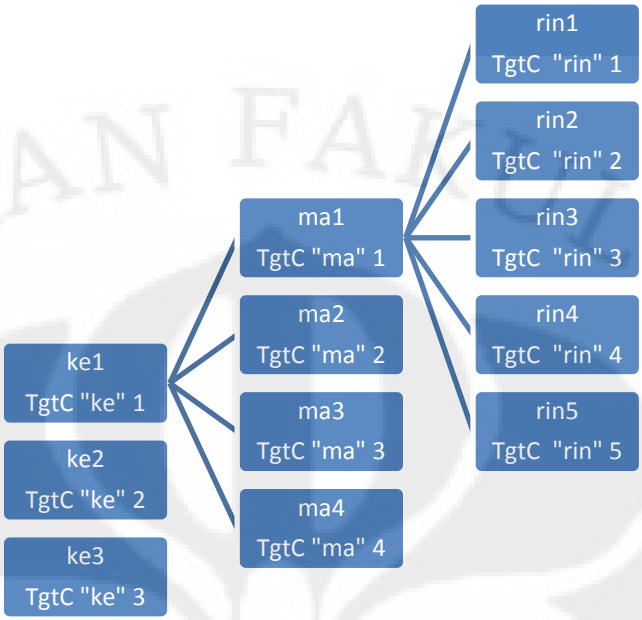
Setelah diperoleh property dari *signal* ucapan suku kata itu, proses berlanjut ke modul Unit Selection Synthesis. Melalui modul ini, nilai dari *target cost* dan *join cost* serta *total cost* akan dihitung.

Parameter untuk perhitungan *target cost* berdasarkan pada *pitch* saja. Nilai *pitch* yang diperoleh dari modul ProsodyPredicting akan dibandingkan dengan nilai *pitch* yang terdapat di dalam *database* dan dihitung *Manhattan Distance*-nya yang merupakan nilai absolute dari selisih keduanya. Nilai jarak ini merupakan nilai cost dari *target cost*.

Sedang untuk *join cost*, parameter yang digunakan berupa *pitch* dan MFCC. Nilai dari *pitch* dan MFCC dari suku kata yang satu dengan nilai *pitch* dan MFCC dari suku kata berikutnya akan ditentukan jaraknya (*distance*) dengan menggunakan system metric *Manhattan Distance.* Nilai dari jarak ini kemudian dijumlahkan dan hasilnya adalah *join cost* untuk kata itu.

Misalnya untuk kata “kemarin”. Kata kemarin memiliki 3 buah suku kata, yaitu “ke”, “ma”, dan “rin”. Katakanlah ketiga suku kata ini berjumlah masing-masing 3, 4, dan 5 di dalam *database* beserta dengan nilai *pitch* dan MFCC yang

berbeda-beda.



Gambar 4.6. Penentuan cost untuk kata "kemarin".

Untuk masing-masing suku kata dicari nilai *target cost*-nya. Untuk suku kata "ke", nilai *pitch* yang diperoleh dari modul ProsodyPredicting dikurangi dengan nilai *pitch* suku kata "ke" yang ada di *database* kemudian ambil nilai mutlak dari hasil selisih tersebut. Karena di *database* terdapat 3 buah suku kata "ke", maka *target cost* untuk suku kata "ke" ada 3 buah. Begitu juga untuk suku kata "ma" dan "rin" sehingga masing-masing memiliki *target cost* 4 buah dan 5 buah.

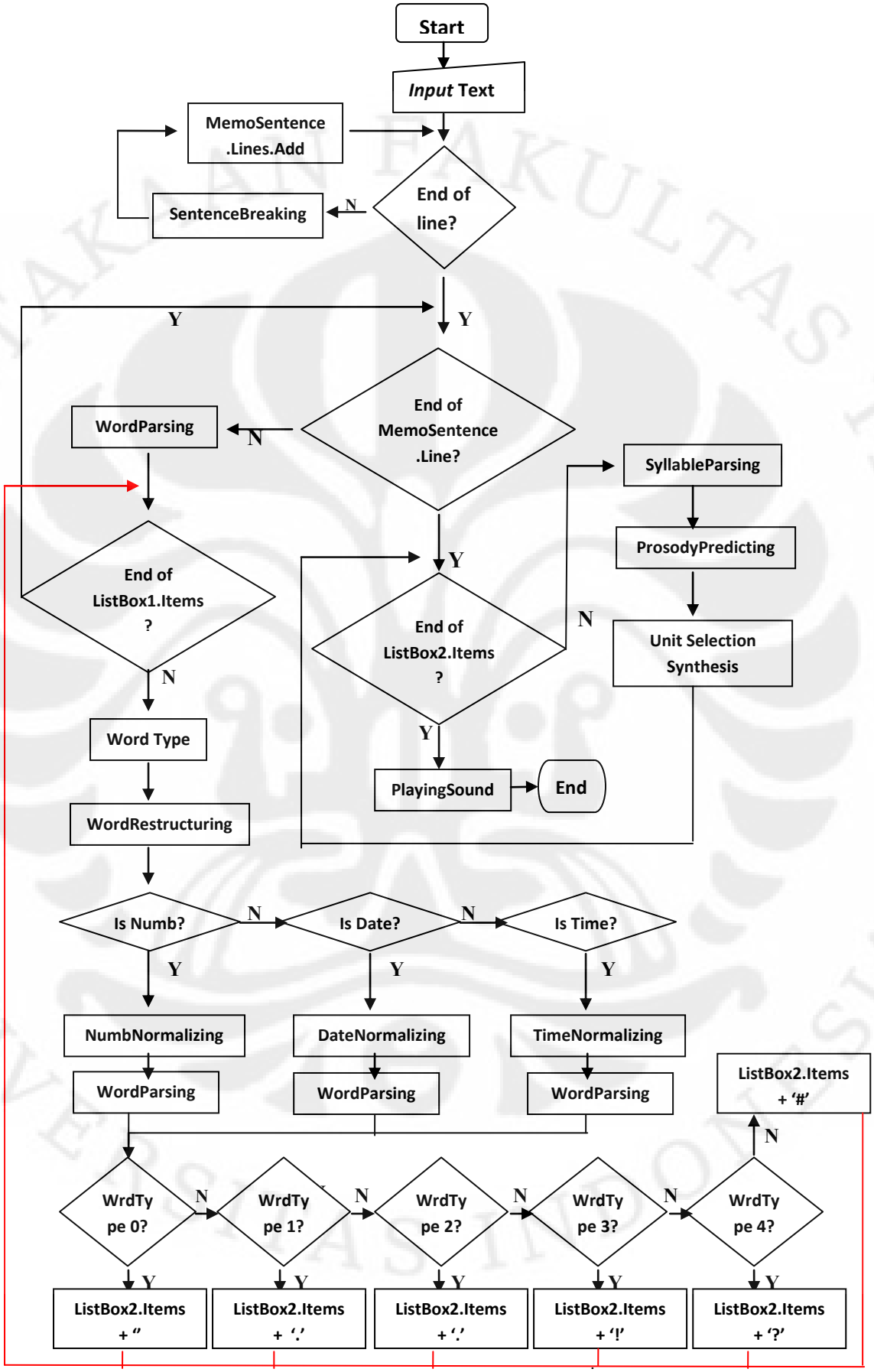
Perhitungan *join cost* dilakukan antara suku kata yang satu dengan suku kata berikutnya. Pada contoh ini, maka *join cost* yang terjadi adalah antara "ke" dan "ma", dan antara "ma" dan "ke". *Join cost* dihitung pada tiap kemungkinan pasangan suku kata: ke1 dengan ma1, ke1 dengan ma2, ma1 dengan rin1, ma1 dengan rin2, dst. Untuk contoh ini, jumlah pasangan yang ada adalah 3x4x5 = 60 pasangan. Sehingga *join cost* yang ada berjumlah 60 buah.

Nilai *total cost* dihitung dengan menjumlahkan antara *target cost* dengan *join cost*. Karena ada 60 pasangan, maka *total cost* berjumlah 60 buah juga. Jadi, nilai dari *total cost* yang pertama adalah *target cost* ke1 + *join cost* (ke1,ma1) + *target cost* ma1 + *join cost* (ma1,rin1). *Total cost* yang kedua adalah *target cost* ke1 + *join cost* (ke1,ma1) + *target cost* ma1 + *join cost* (ma1,rin2), dst.

Diantara keenam puluh nilai *total cost* yang ada, dicari nilai yang paling kecil. Urutan pasangan suku kata yang membentuk kata "kemarin" dengan nilai *total cost* yang paling kecil dapat diperoleh dengan mencari pada urutan keberapakah nilai *total cost* yang paling kecil tersebut.

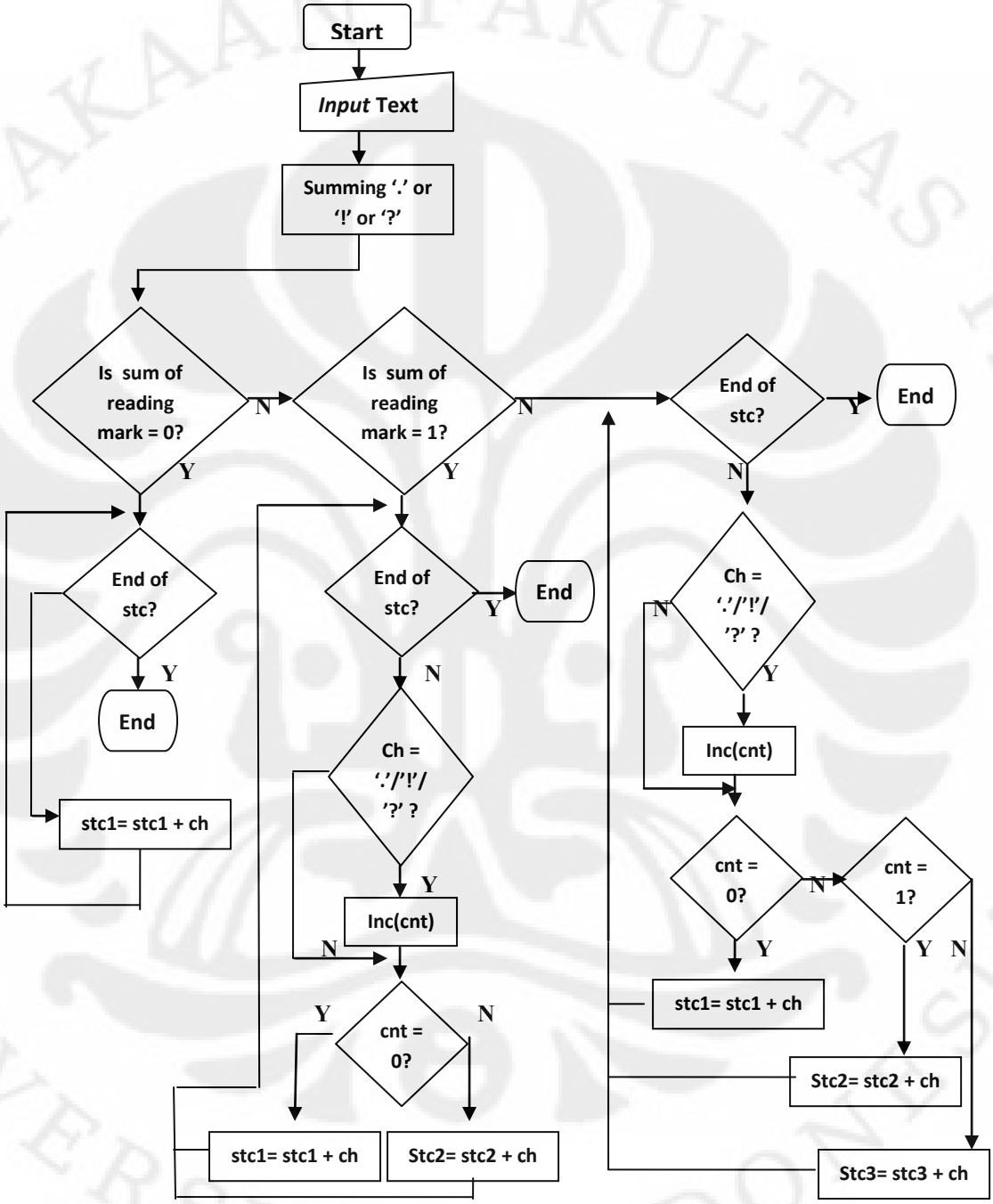
Pada contoh ini, setelah ditemukan bahwa nilai yang paling kecil diantara sederet *total cost* itu berada pada urutan ke 4 –misalnya- maka pasangan suku katanya adalah ke1, ma1, dan rin4.

Flowchart yang menggambarkan algoritma dari modul utama dapat dilihat pada Gambar 4.7 di bawah ini:


Start
Input Text
MemoSentence .Lines.Add
End of line?
N
SentenceBreaking
Y
Y
End of MemoSentence .Line?
N
WordParsing
Y
End of ListBox1.Items ?
N
Word Type
WordRestructuring
End of ListBox2.Items ?
N
Y
SyllableParsing
ProsodyPredicting
Unit Selection Synthesis
PlayingSound
End
Is Numb?
N
Is Date?
N
Is Time?
Y
Y
Y
NumbNormalizing
DateNormalizing
TimeNormalizing
WordParsing
WordParsing
WordParsing
WrdTy pe 0?
N
WrdTy pe 1?
N
WrdTy pe 2?
N
WrdTy pe 3?
N
WrdTy pe 4?
N
ListBox2.Items + '#'
Y
Y
Y
Y
Y
ListBox2.Items + ''
ListBox2.Items + '.'
ListBox2.Items + '.'
ListBox2.Items + '!'
ListBox2.Items + '?'

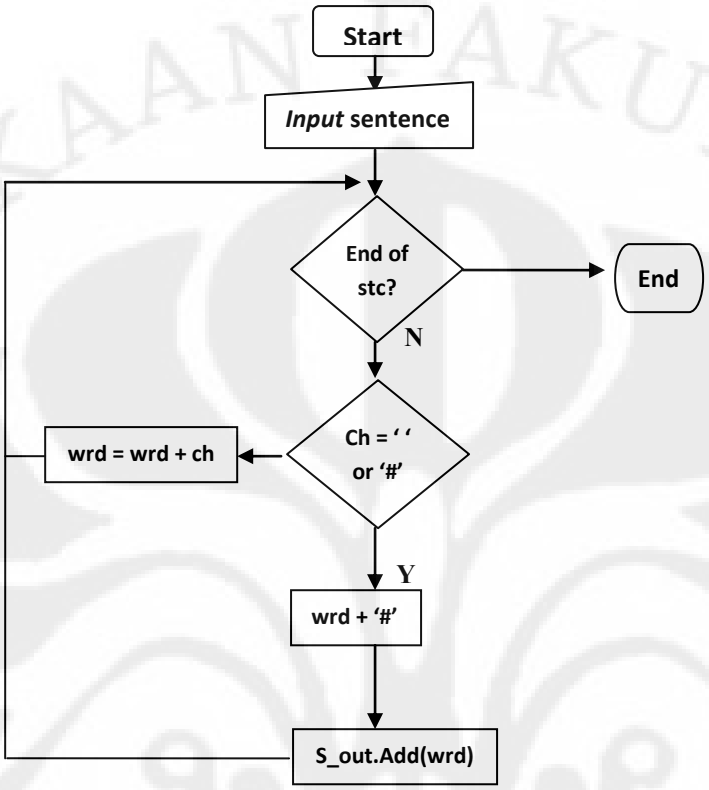
Gambar 4.7. Flowchart modul utama.

Untuk flowchart dari algoritma modul SentenceBreaking dapa dilihat pada Gambar 4.8 di bawah ini:



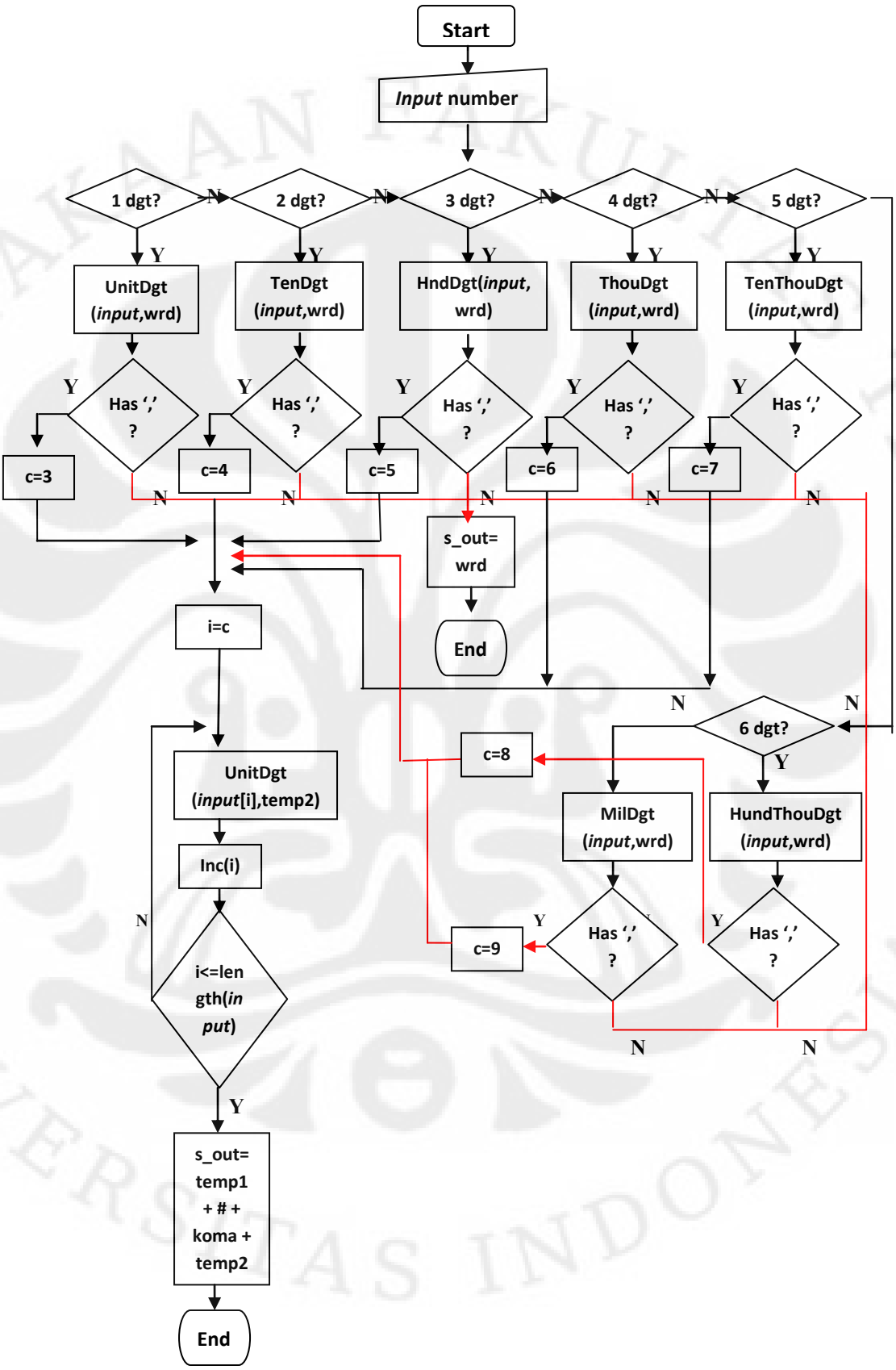
Gambar 4.8. Flowchart untuk modul SentenceBreaking.

Flowchart untuk algoritma dari modul WordParsing secara garis besarnya dapat dilihat pada Gambar 4.9 di bawah ini:



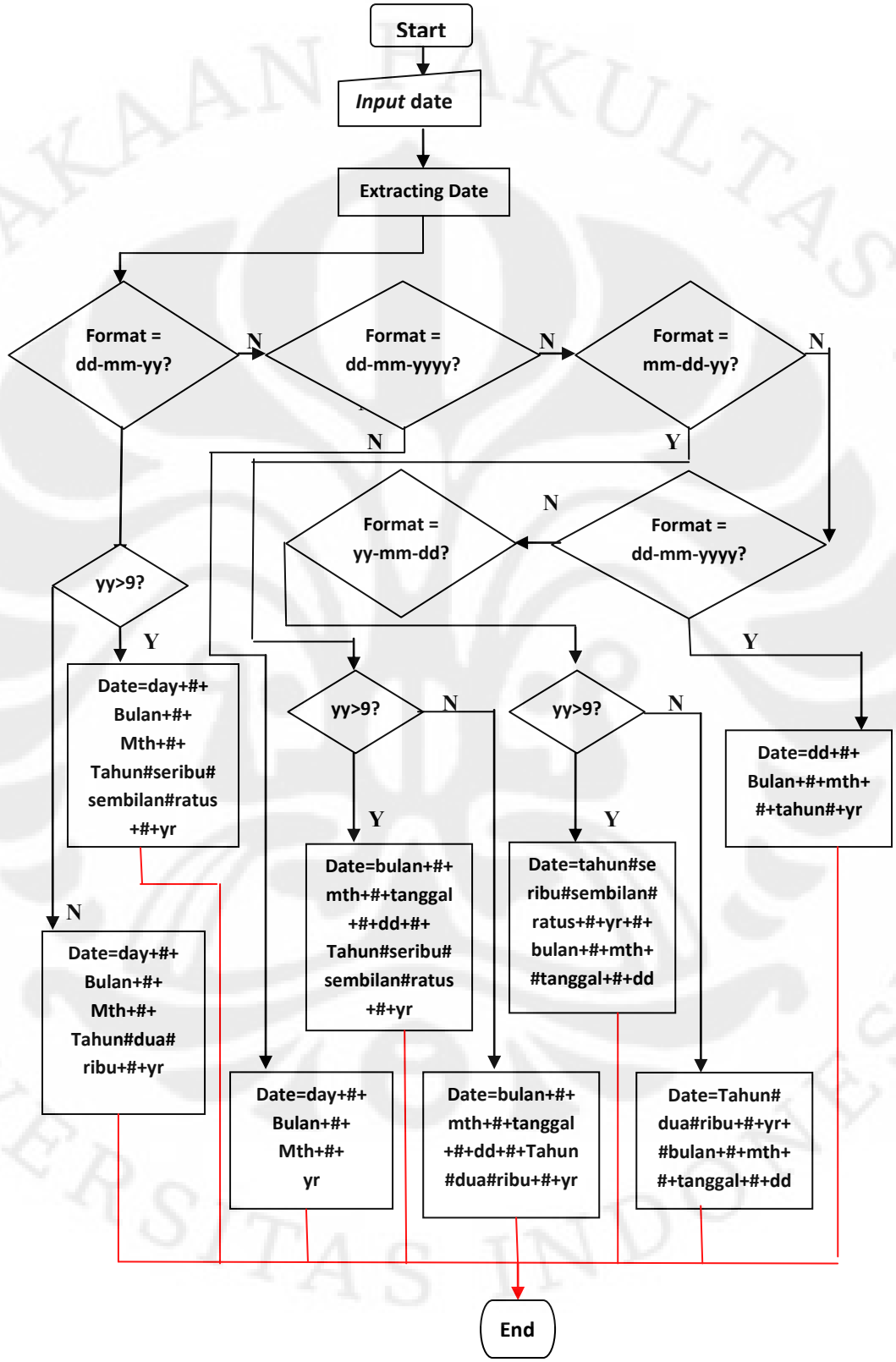
Gambar 4.9. Flowchar modul WordParsing.

Flowchar dari modul NumbNormalizing dapat dilihat pada Gambar 4.10 di bawah ini:



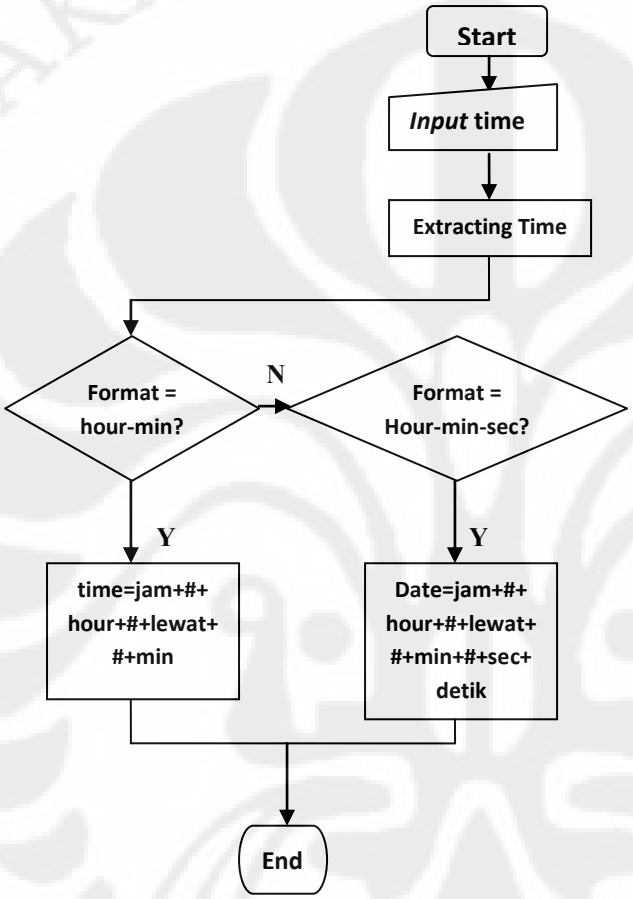
Gambar 4.10. Flowchart modul NumbNormalizing.

Flowchart untuk modul DateNormalizing dapat dilihat pada Gambar 4.11 di bawah ini:



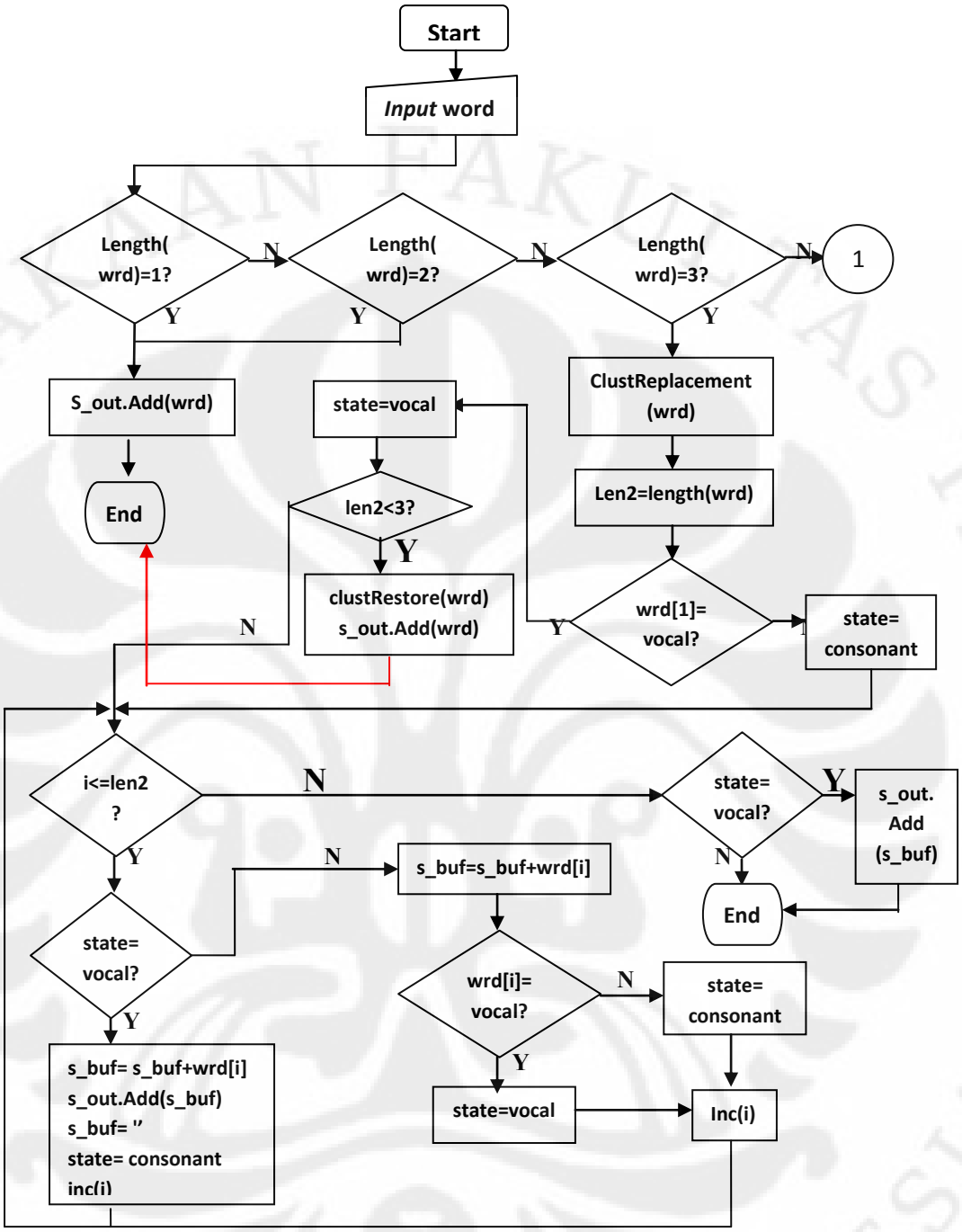
Gambar 4.11. Flowchart modul DateNormalizing.

Untuk flowchart dari modul TimeNormalizing dapat dilihat pada Gambar 4.12 di bawah ini:

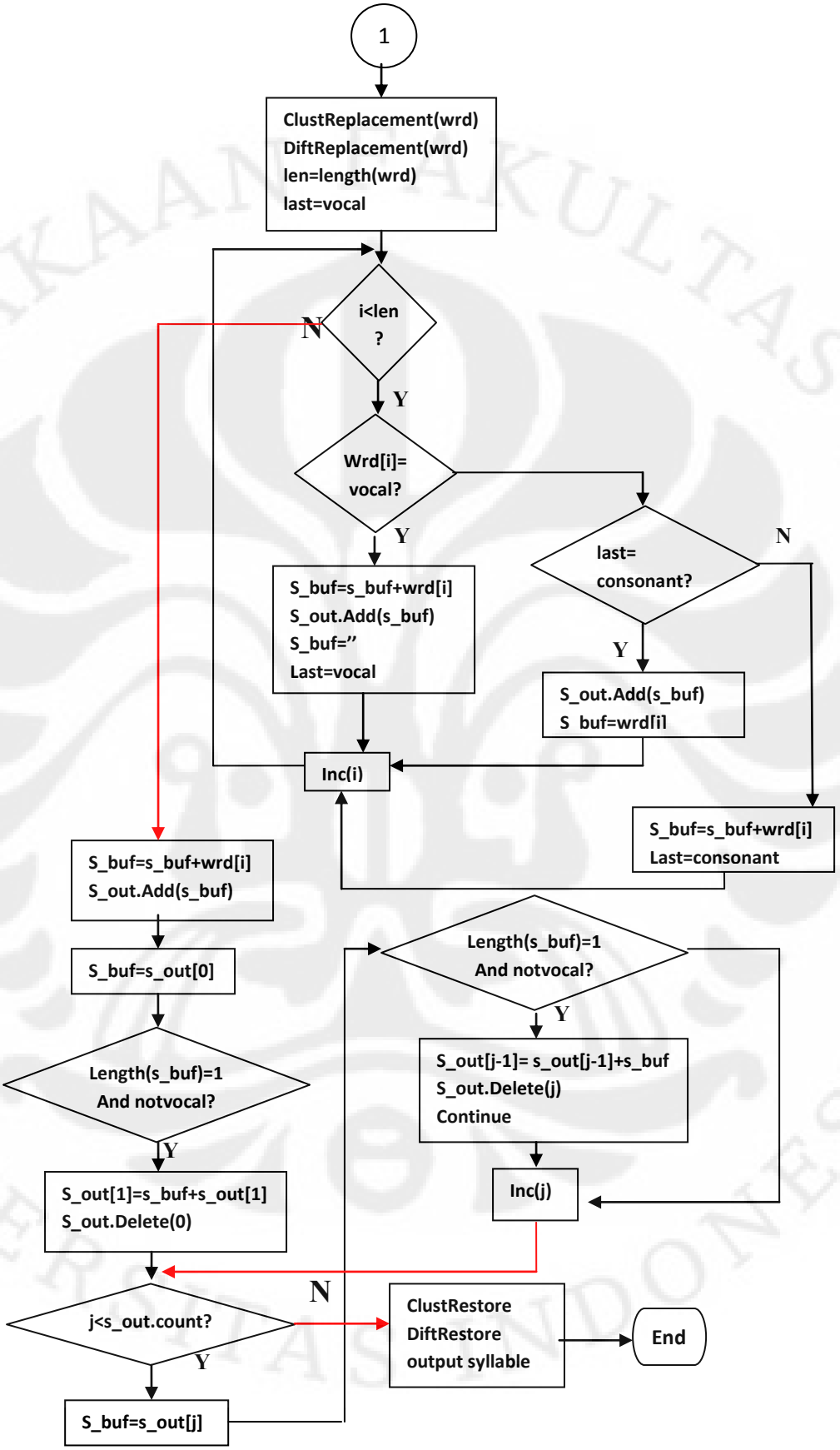


Gambar 4.12. Flowchart modul TimeNormalizing.

Untuk flowchart modul SyllableParsing dapat dilihat pada Gambar 4.13 di bawah ini:



Gambar 4.13. Flowchart modul SyllableParsing.

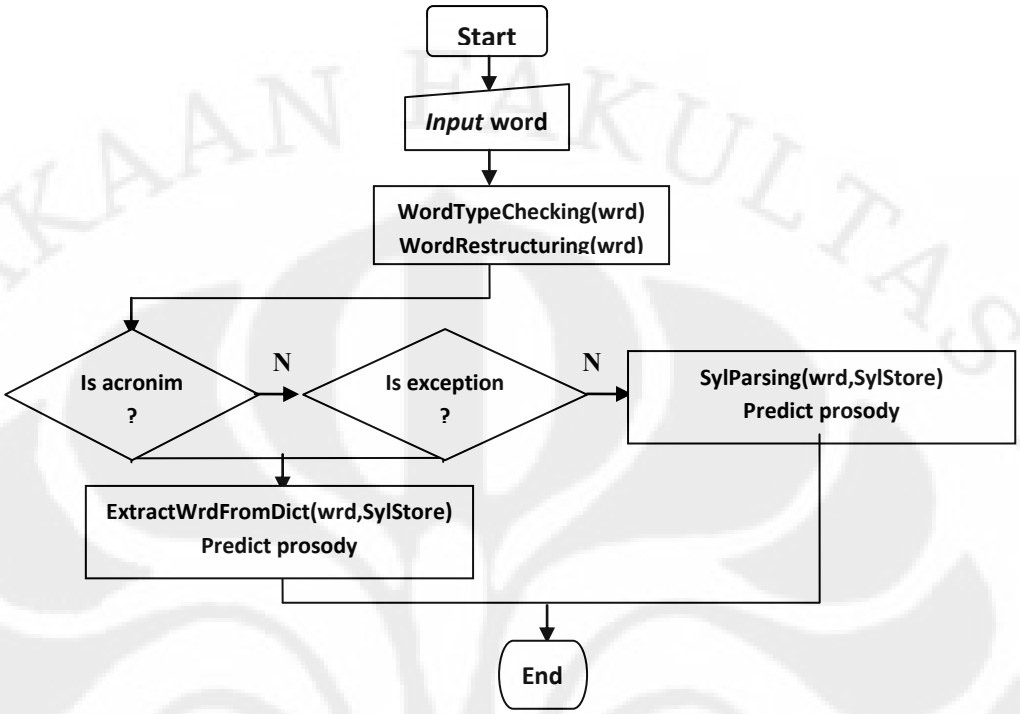


Gambar 4.13. Flowchart untuk modul Syllable Parsing (lanjutan).

Inti dari algoritma dari modul SyllableParsing adalah mengeluarkan huruf atau gabungan huruf-huruf bila huruf tersebut dalah vocal atau berakhiran vocal. Misalnya dari kata "menguburkan". Tahapan dari proses pemecahan suku katanya pertama kali adalah dengan mengganti kluster (gabungan 2 konsonan) dengan huruf lain yang selain abjad (pada skripsi ini, "ng" diganti dengan "\"). Sehingga kata yang baru menjadi "me\uburkan". Kemudian dilakukan parsing sehingga menghasilkan suku kata "me", "\u", "bu", "r", dan "kan" (karena suku kata "kan" merupakan suku kata yang terakhir, maka suku kata "ka" dan konsonan "n" akan digabung menghasilkan "kan" setelah proses iterasi di dalam prosedur berakhir). Bila ditemui ada satu buah konsonan di dalam pemecahan tersebut, maka satu konsonan itu akan digabungkan dengan suku kata sebelumnya. Sehingga proses selanjutnya menghasilkan suku kata "me", "\u", "bur", dan "kan". Langkah selanjutnya adalah mengembalikan pertukaran kluster ataupun diftong (gabungan 2 buah vocal) menjadi seperti semula sehingga hasil berikutnya adalah suku kata "me", "ngu", "bur", dan "kan". Kemudian langkah terakhir adalah tinggal mengeluarkan suku kata tersebut.

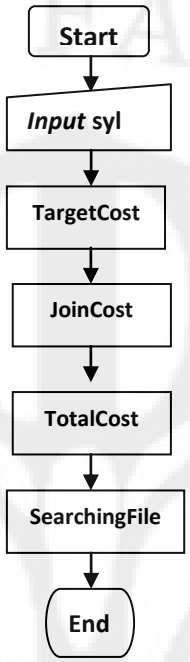
Contoh lainnya, misalnya kata "teknik". Setelah mengalami parsing tahap pertama, menghasilkan suku kata "te", "k", dan "nik". Karena terdapat suku kata yang memiliki satu buah konsonan, maka konsonan tersebut akan digabung dengan suku kata sebelumnya sehingga menghasilkan suku kata "tek", dan "nik". Alhasil, suku kata tersebut tinggal dikeluarkan.

Flowchart untuk modul ProsodyPredicting dapat dilihat pada Gambar 4.14 di bawah ini:



Gambar 4.14. Flowchart modul ProsodyPredicting.

Flowchart untuk modul UnitSelectionSynthesis dapat dilihat pada Gambar 4.15 di bawah ini:



Gambar 4.15. Flowchart modul UnitSelectionSynthesis.

Pada prosedur TargetCost, masukan yang berupa suku kata beserta dengan prosodinya akan digunakan untuk melakukan penyaringan (*filtering*) *database*. Suku kata beserta dengan prosodi (properti prosodi dalam hal ini adalah berupa *accent* dan *break*) yang ada di dalam *database* yang memiliki kesamaan dengan *input* akan dipilih. Lalu nilai *pitch* yang merupakan hasil proses dari modul ProsodyPredicting dan nilai *pitch* yang ada di dalam *database* akan diproses untuk mendapatkan nilai *Manhattan distance*-nya untuk setiap suku kata yang telah mengalami filterisasi. Nilai-nilai Manhattan distance yang diperoleh akan dimasukkan ke dalam array 2 dimensi yang mana dimensi pertama menyatakan indeks dari urutan suku kata dari kata asli yang sedang diproses dan dimensi kedua menyatakan urutan dari suku kata yang sama dengan *input* ada di dalam *database* setelah mengalami pengurutan. Misalnya dari kata "kemarin", bila nilai dari dimensi pertama adalah 2 dan dimensi ke 2 bernilai 3. Dan anggap di dalam *database* terdapat 3 buah suku kata "ke", 4 buah suku kata "ma" dan 5 buah suku kata "rin". Dimensi pertama menyatakan bahwa suku kata yang sedang diproses

adalah suku kata kedua dari kata "kemarin", yaitu suku kata "ma". Dimensi kedua menyatakan bahwa suku kata "ma" yang ada di *database* yang diproses adalah suku kata "ma3". Sehingga, nilai dari *Manhattan distance* adalah selisih dari nilai *pitch* suku kata "ma" yang merupakan hasil keluaran modul ProsodyPredicting dengan nilai *pitch* dari suku kata "ma3" yang ada di dalam *database*.

Prinsip yang sama diberlakukan pada prosedur JoinCost. Hanya saja, array yang digunakan adalah array berdimensi 3. Penjelasan dari ketida dimensi tersebut akan dijelaskan melalui contoh berikut ini. Misalnya masih menggunakan contoh kata yang sama, yaitu "kemarin". Dan di *database* terdapat 3 buah suku kata "ke", 4 buah suku kata "ma", dan 5 buah suku kata "rin". Masing-masing suku kata tersebut memiliki *file wav*-nya sendiri-sendiri yang disimpan di dalam suatu folder tertentu (di dalam skripsi ini disimpan di dalam folder bernama **eka** karena suara dari *database* yang dibuat dari seseorang yang bernama Eka Darmaputera), eka\ke1.wav, eka\ke2.wav, eka\ke3.wav, eka\ma1.wav, dst. Susunan data di dalam *database* disusun berdasarkan nama dari *file wav* tersebut. Karena dalam perhitungan *join cost* terdapat perhitungan antara suku kata yang satu dengan suku kata berikutnya, maka pasangan suku kata yang terjadi dalam perhitungan *join cost* untuk kata "kemarin" adalah ke-ma, dan ma-rin.

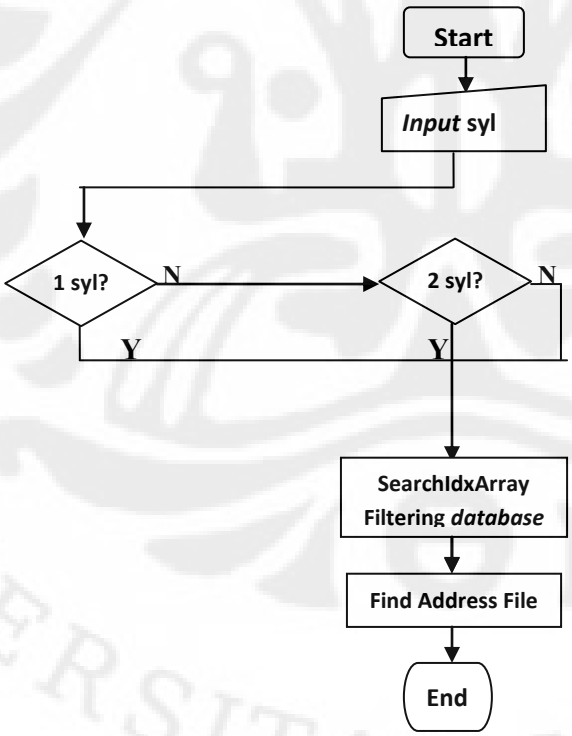
Pada pasangan suku kata yang pertama (ke-ma), proses perhitungan *join cost* adalah sebagai berikut. Dimensi pertama dari array *join cost* menyatakan urutan suku kata yang terdapat di dalam kata "kemarin". Jadi, bila dimensi pertama ini bernilai 1, maka urutan suku kata yang dimaksud menunjuk kepada *file* suku kata **"ke"**. Dimensi kedua dari array *join cost* menyatakan urutan suku kata yang ada di dalam *database* untuk suku kata yang ditunjuk oleh dimensi pertama (untuk kasus ini, suku kata yang dimaksud adalah suku kata "ke"). Jadi, bila dimensi kedua ini bernilai 3, maka suku kata di dalam *database* yang dimaksud adalah suku kata "ke" dengan *file wav* bernama **eka\ke3.wav**. Dimensi ketiga dari array *join cost* menyatakan urutan suku kata berikutnya yang ada di dalam *database* (untuk kasus ini berarti suku kata yang dimaksud adalah suku kata "ma"). Bila dimensi ketiga ini bernilai 1, maka suku kata ma yang dimaksud adalah suku kata ma yang memiliki *file wav* bernama **eka\ma1.wav**. Sebagai contoh lagi, bila array *join cost* berkoordinat [1,2,3], maka koordinat ini

menyatakan perhitungan *join cost* antara suku kata "ke" dengan atribut *pitch*, ToBI, dan MFCC yang terdapat pada **ke2.wav** dengan suku kata berikutnya, yaitu suku kata "ma" dengan atribut yang terdapat pada **ma3.wav**.

Perhitungan *join cost* dilakukan untuk semua kemungkinan pasangan yang terdapat di dalam *database* untuk suatu kata yang bersangkutan. Untuk contoh yang tadi, maka semua kemungkinan yang ada berjumlah 3x4x5 = 60 kemungkinan.

Untuk perhitungan *target cost* dan *join cost*, nilai terbobot (*weighted value*) dianggap bernilai 1, sehingga tidak dilakukan training data.

Pada perhitungan *total cost*, nilai dari *target cost* dengan *join cost* dijumlahkan kemudian hasilnya disimpan di dalam array 1 dimensi. Nilai-nilai yang terdapat di dalam array *total cost* ini kemudian diurutkan tanpa diubah posisi dari nilai tersebut di dalam array, kemudian dicari nilai terkecilnya. Nilai terkecil yang terdapat di dalam array itu dicari indeksnya oleh modul SearchingFile. Berikut adalah flowchart dari modul SearchingFile:



Gambar 4.16. Flowchart dari modul SearchingFile.

Cara untuk menentukan *file wav* suku kata mana yang diinginkan dapat dilihat pada potongan sintaks program yang dilampirkan pada LAMPIRAN A.

Penjelasan parameter dari prosedur tersebut adalah sebagai berikut:

1. SylTotal : jumlah suku kata
2. syl1 : jumlah unit di *database* untuk suku kata pertama dari kata *input*an.
3. syl2 : jumlah unit di *database* untuk suku kata kedua dari kata *input*an.
4. syl3 : jumlah unit di *database* untuk suku kata ketiga dari kata *input*an.
5. n : index elemen untuk nilai terkecil dari array *total cost*.
6. x : index untuk menentukan *file wav* suku kata pertama dari kata *input*an.
7. y : index untuk menentukan *file wav* suku kata kedua dari kata *input*an.
8. z : index untuk menentukan *file wav* suku kata ketiga dari kata *input*an.

Setelah index dari suku kata telah ditentukan, index-index tersebut akan digunakan oleh prosedur **Save2Mem** untuk menyimpan *file-file wav* suku kata tadi ke dalam suatu memori. Lalu *file* suku kata yang telah telah di-pre-load tadi ke dalam memori kemudian akan dimainkan secara keseluruhan oleh prosedur **PlayingSound**.

BAB V
IMPLEMENTASI DAN ANALISA SISTEM

5.1. Implementasi Sistem

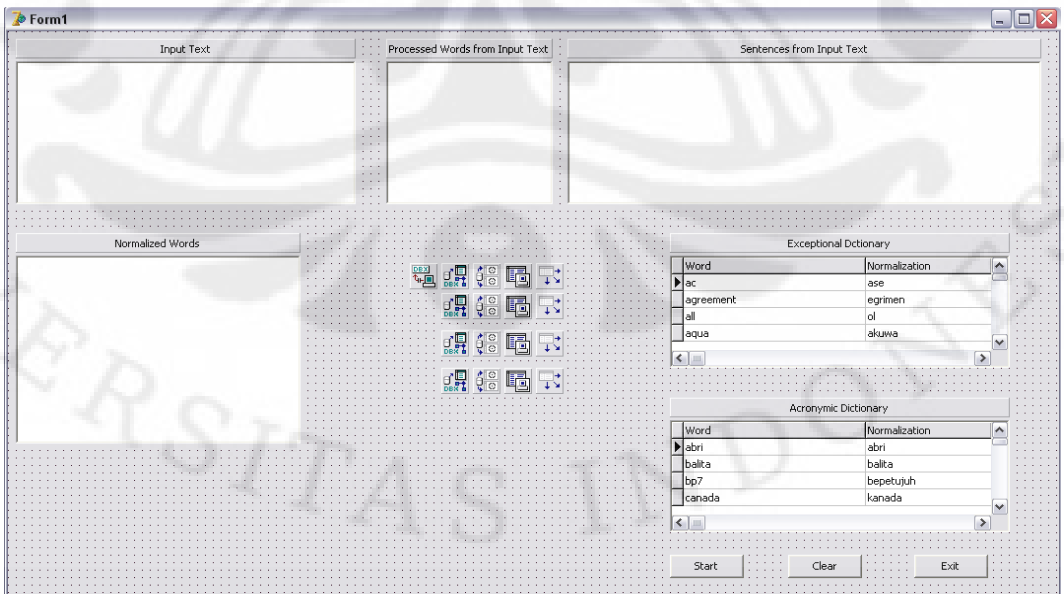
Implementasi dari system TTS yang dibuat pada skripsi ini dilakukan pada Laptop dengan spesifikasi *hardware* sebagai berikut:

- Merk : Acer ASPIRE 4530
- Processor : AMD-Turion X2 RM-70 2.00 GHz
- RAM : DDR2 766 MB
- Audio device : Realtek HD Audio *Output*

Sedang software yang digunakan adalah:

- Programming software : Borland Delphi 7 Enterprise Edition
- *Database* : MySQL Front v 5.0
- Programming Language : Pascal
- Third Party software : WavePad Sound Editor v 4.27 dan Praat v 5.1.07

Adapun komponen visual yang digunakan dalam memrogram system pada Delphi 7 dapat dilihat pada Gambar 5.1 di bawah ini:



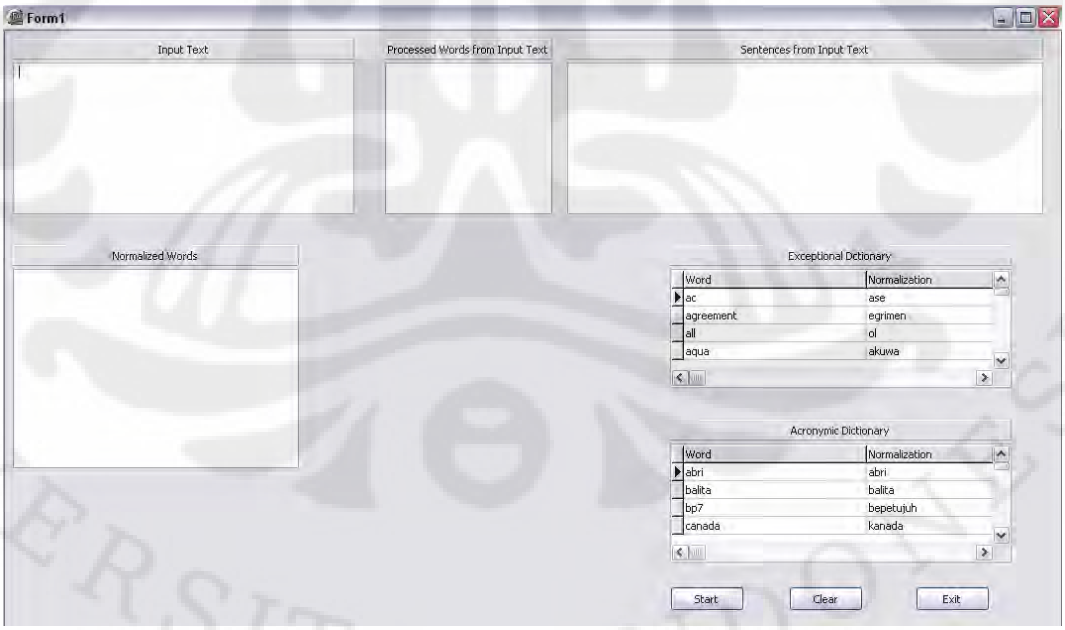
Gambar 5.1. Komponen visual yang digunakan.

Komponen-komponen tersebut antara lain:

Tabel 5.1. Daftar Komponen

No	Komponen	No	Komponen
1	Memo*Input*	17	Panel6
2	ListBox1	18	DataSetProvider1
3	Button1	19	ClientDataSet1
4	ListBox2	20	DataSource1
5	SQLConnection1	21	MemoSentence
6	SQLDataSetExcp	22	Panel7
7	SQLDataSetAcr	23	Button2
8	DataSetProvider2	24	Button3
9	ClientDataSet2	25	DataSetProvider3
10	DataSource2	26	ClientDataSet3
11	DBGrid	27	SQLDataSetSyllable
12	Panel1	28	DataSource3
13	Panel2	29	DataSetProvider4
14	Panel3	30	ClientDataSet4
15	DBGrid2	31	SQLDataSetMeanPitch
16	Panel5	32	DataSource4

Sedang tampilan GUI system TTS ini dapat dilihat pada Gambar 5.2 di bawah ini:



Gambar 5.2. Tampilan GUI dari system TTS

Input teks yang ingin diproses diketik pada memo yang berjudul Input Text. Banyaknya kalimat maksimal yang dapat dimasukkan pada memo ini untuk

1 baris adalah 3 kalimat saja. Bila lebih dari 3 kalimat, maka sistem tidak dapat dijalankan. Sistem ini masih dapat memroses kalimat yang tidak utuh asalkan kelanjutan dari kalimat tersebut dilanjutkan pada baris selanjutnya.

Analisis yang dilakukan mencakup 3 hal: analisis waktu, analisis *file wav* yang diambil, analisis intellijibilitas dan kealamian ucapan, dan analisis *signal* ucapan berikut dengan atributnya.

5.2. Analisis waktu.

Analisis waktu yang dimaksud di sini adalah analisis terhadap waktu yang diperlukan oleh sistem ketika mengucapkan suatu kata dibandingkan dengan waktu yang ada di dalam *database*. Selain itu analisis juga mencakup waktu jeda yang terjadi antar kata.

Kalimat masukan yang dijadikan sebagai bahan analisis adalah : **"dodi yang tadi yang ingat 368."**. Waktu total yang dibutuhkan oleh program untuk mengucapkan kalimat ini mulai dari ditekannya tombol start adalah sekitar **7.60 detik**. Sedang waktu yang diperlukan untuk mengucapkan kalimatnya saja adalah sekitar **4.09 detik**. Hal ini berarti waktu yang dibutuhkan oleh program untuk melakukan proses dari awal sampai penyimpanan *file wav* untuk kalimat ini ke memori adalah sekitar **3.51 detik**. Dengan demikian, program yang dibuat dapat dikatakan masih kurang efisien. Data dari suku kata yang dipakai beserta dengan atributnya adalah sebagai berikut:

Tabel 5.2. Data percobaan 1.

Kata	Suku kata			Pitch (Hz)			Durasi (ms)			File Wav			Durasi	Ttl Cost
	1	2	3	1	2	3	1	2	3	1	2	3	ttl(ms)	
dodi	do	di		127	190		137	127.17		do1	di1		264.17	109.71
yang	yang			187			318.67			yang1			318.67	0.22
tadi	ta	di		127	163		129	127.17		ta3	di1		256.17	131.79
yang	yang			184			318.67			yang1			318.67	2.78
ingat	i	ngat		153	199		80.67	238.17		i1	ngat1		318.84	258.22
tiga	ti	ga		194	291		93.17	117.5		ti2	ga2		210.67	136.52
ratus	ra	tus		195	200		157.17	256.5		ra1	tus1		413.67	100.4
enam	e	nam		184	167		52.67	361.5		e1	nam1		414.17	266.58
puluh	pu	luh		164	175		91.33	147.83		pu1	luh1		239.16	156.8
delapan	de	la	pan	164	188	181	84	212.33	260.17	de1	la2	pan1	296.33	278.31
													3050.5	1441.3

Berdasarkan hasil data percobaan di Tabel 5.2, durasi total pengucapan kalimat berdasarkan penjumlahan durasi tiap suku kata adalah 3050,5 ms atau **3.05 detik**. Ini adalah durasi yang semestinya terjadi. Namun, hasil dari percobaan

adalah 4.09 detik. Ini berarti terjadi keterlambatan sekitar **1.04 detik** untuk pengucapan kalimat ini. Keterlambatan ini terjadi karena diperlukannya waktu oleh komputer untuk me-*load file* suku kata yang telah disimpan terlebih dahulu ke memori. Jumlah suku kata total pada kalimat ini adalah 19 suku kata. Sehingga terdapat 19 kali pemanggilan *command* PlaySound. Ini berarti jeda antara suku kata yang satu dengan suku kata berikutnya adalah sekitar

Delay = 1.04 : 18

57.78 ms

Antara suku kata yang satu dengan suku kata berikutnya semestinya diberikan jeda. Namun, pada program ini tidak diberikan jeda antar suku kata kecuali jika pada akhir dari suku kata tersebut terdapat tanda baca (titik, koma, tanda tanya, dan tanda seru). Bila tanda baca berupa tanda titik, tanda tanya, atau tanda seru, maka jeda yang diberikan berdurasi 30 ms. Bila tanda baca berupa koma, maka jeda yang diberikan berdurasi 20 ms. Bila tidak ada tanda baca sama sekali, maka tidak diberikan jeda.

Meskipun tidak diberikan jeda antar suku kata, namun pada percobaan tetap terdengar adanya jeda. Hal ini disebabkan karena pada waktu pembuatan *database*, ketika melakukan pemotongan gelombang suara dari *file* yang dijadikan sumber data, jeda yang terjadi yang memang dari awalnya sudah ada di dalam *file* tersebut, juga diikutsertakan. Sehingga *file* suku kata yang terbentuk sudah mengandung jeda. Selain jeda yang telah terkandung di dalam *file* suku kata, jeda juga terjadi ketika pemanggilan *file* dilakukan. Dengan demikian, durasi jeda minimum yang terjadi antar suku kata yang tidak memiliki tanda baca adalah 57.78 ms. Oleh karena itu, jeda antar suku kata yang tidak memiliki tanda baca tidak diberikan. Bila diberikan justru membuat kalimat yang diucapkan akan terdengar kurang alami.

5.3. Analisis *File wav* yang diambil

Dari data percobaan pada Tabel 5.2, dapat dilihat bahwa *file wav* yang diambil sudah sesuai dengan program yang dibuat. Untuk kata pertama pada kalimat percobaan (“dodi”), suku kata yang diambil adalah “do1” dan “di1”. Suku kata tersebut memang sudah pasti yang akan dipilih oleh program terlepas dari

berapapun nilai *cost* yang dihasilkan karena suku kata "do" dan "di" baru ada masing-masing satu buah pada *database*. Suku kata do memiliki atribut *break* 0 yang menandakan bahwa suku kata ini bukan suku kata akhir suatu kata. Dan suku kata "di" memiliki atribut *break* "1" yang menandakan bahwa suku kata ini merupakan suku kata akhir dari suatu kata. Atribut accent dari kedua suku kata tersebut adalah "L" yang menandakan bahwa suku kata itu tidak dapat dipakai untuk akhir suatu kalimat utuh dan tidak dapat dipakai untuk kata yang memerlukan tanda koma sesudahnya. Suku kata "do" dan "di" tadi sudah sesuai dengan persyaratan yang diajukan sebab pada kata "dodi" di kalimat percobaan tadi merupakan kata yang tidak memerlukan tanda koma dan bukan merupakan akhir dari kalimat utuh. Hal yang serupa juga terjadi untuk kata "ratus", "enam", "puluh", dan "delapan". Kata-kata tersebut memiliki suku kata berikut dengan karakteristiknya yang khusus yang masing-masing masih berjumlah satu di dalam *database*.

Kasus yang berbeda terjadi untuk suku kata "yang" (yang juga merupakan suatu kata pada urutan ke 2 dan ke 4 di kalimat percobaan). Suku kata yang dipilih oleh program adalah suku kata "yang1". Suku kata ini memiliki atribut ToBI *break* "1" yang menandakan bahwa suku kata ini merupakan suku kata akhir dari suatu kata. Atribut accent dari suku kata ini adalah "L" yang menandakan bahwa suku kata ini bukan merupakan akhir dari suatu kalimat utuh dan tidak memerlukan tanda koma. Dengan demikian, program telah memilih suku kata "yang" dengan atribut yang sesuai dengan kalimat percobaan, sebab kata "yang" pada kalimat percobaan bukan merupakan kata akhir dari kalimat utuh dan juga tidak memerlukan tanda koma. Suku kata yang dipilih adalah suku kata "yang1" sebab suku kata inilah yang memiliki nilai *total cost* yang paling kecil dibandingkan dengan suku kata "yang" yang lain yang ada di dalam *database*. Hal ini dapat dilihat dari *pitch* yang dihasilkan oleh *prosody generator*. Nilai *pitch* tersebut adalah 187 Hz untuk kata "yang" yang pertama dan 184 Hz untu kata "yang" yang kedua. Karena kata "yang" di sini terdiri dari satu suku kata saja, maka *total cost* bernilai sama dengan nilai *target cost*. Nilai *target cost* untuk kata "yang" ini adalah selisih dari nilai *pitch* hasil dari prosody generator dengan nilai *pitch* yang ada di *database*. Nilai *pitch* untuk kata "yang1" adalah 187 sehingga

nilai *pitch* yang terdekat dengan nilai *pitch* ini adalah 186.78 Hz, yaitu nilai *pitch* yang dimiliki oleh suku kata "yang1". Dengan demikian, nilai *total cost* adalah 187 – 186.78 = 0.22. Suku kata "yang" yang berikutnya memiliki nilai *pitch* keluaran dari prosody generator adalah 184 Hz. Nilai *pitch* terdekat dengan nilai *pitch* ini adalah nilai *pitch* yang dimiliki oleh suku kata "yang1", yaitu 186.78 Hz. Dengan demikian, nilai dari *total cost* untuk suku kata "yang" yang kedua ini adalah 186.78-184 = 2.78. Program dengan demikian bekerja dengan baik, yaitu berhasil dalam memilih suku kata "yang" dengan nilai *total cost* yang terendah diantara ketiga suku kata "yang" yang lain yang ada dalam *database*.

Pada kata "tadi", suku kata yang dipilih adalah "ta3" dan "di1". Suku kata "di1" mau tidak mau harus dipilih oleh program karena suku kata ini dengan atributnya yang sesuai dengan kalimat adalah satu-satunya suku kata yang ada di dalam *database*. Tidak demikian halnya dengan suku kata "ta". Meskipun suku kata "ta3" memiliki nilai *pitch* yang paling besar diantara suku kata "ta" yang lain, namun tetap suku kata "ta3" ini yang dipilih oleh program. Suku kata "ta3" memang memiliki nilai *target cost* yang paling tinggi (karena memiliki nilai *pitch* yang paling tingg), namun, nilai dari *pitch* MFCC suku kata "ta3" ini memiliki tingkat kedekatan yang paling baik dengan suku kata "di1" dibandingkan dengan suku kata "ta" yang lain. Kedekatan ini menghasilkan nilai *join cost* yang paling rendah sehingga bila dijumlahkan dengan *target cost* dari suku kata "ta3", maka *total cost* yang dihasilkan adalah yang paling kecil daripada suku kata "ta" yang lainnya. Oleh karenanya, suku kata "ta3" yang dipilih. Hal ini sesuai dengan program yang dibuat, yaitu memilih pasangan suku kata dengan nilai *total cost* yang paling rendah. Dengan demikian, program berjalan dengan baik.

Untuk kata "ingat", suku kata yang dipilih adalah suku kata "i1" dan "ngat1". Suku kata "ngat1" sudah jelas akan dipilih oleh program karena memang suku kata "ngat1" ini masih berjumlah satu buah di dalam *database* dan atribut yang dimilikinya sesuai dengan ketentuan dari kalimat. Suku kata "i1" dipilih oleh program karena memiliki *target cost* yang lebih kecil dibandingkan dengan *target cost* yang dimiliki oleh suku kata "i2". Selain itu, suku kata "i1" ini memiliki tingkat kedekatan dengan suku kata berikutnya, yaitu "ngat" yang lebih baik dibandingkan dengan suku kata "i2". Hal ini dapat dilihat dari nilai *pitch* dan

MFCC yang dimiliki oleh suku kata "i1" lebih dekat nilainya dengan nilai *pitch* dan MFCC yang dimiliki oleh suku kata "ngat". Dengan demikian, nilai *target cost* dan *join cost* sama-sama menghasilkan nilai yang paling kecil bila dibandingkan dengan nilai *target cost* dan *join cost* yang dimiliki oleh suku kata "i2".

Pada kata "tiga", suku kata yang dipilih oleh program adalah suku kata "ti2" dan "ga2". Suku kata "ti2" dipilih oleh program karena atribut suku kata "ti" yang sesuai dengan ketentuan yang ada di kalimat membutuhkan atribut *break* "0" karena suku kata "ti" ini bukan suku kata akhir dari suatu kata dan atribut accent "L" karena suku kata "ti" bukan merupakan akhir dari suatu kalimat dan tidak sedang membutuhkan tanda koma. Oleh karena itu, ketika terjadi penyaringan *database* (proses *filtering*), suku kata "ti1" akan tereliminasi karena tidak sesuai dengan syarat yang diinginkan sehingga suku kata "ti" yang tersisa adalah suku kata "ti2" terlepas dari berapapun nilai *total cost* yang dihasilkan. Kasus yang mirip juga dialami oleh suku kata "ga". Sehingga suku kata "ga" yang dipilih adalah suku kata "ga2" terlepas dari berapapun nilai *total cost* yang dihasilkan. Dengan demikian, dapat dikatakan bahwa program dapat melakukan proses penyaringan data dengan baik.

5.4. Analisis intellijibilitas (*intelligibility*) dan tingkat kealamian (*naturalness*) dari ucapan yang dihasilkan oleh system.

Meskipun sistem telah berhasil dalam melakukan pemilihan data *file wav* yang ada di *database* dengan baik, namun kualitas dari ucapan yang dihasilkan tidak baik. Dari segi intellijibilitas, kualitasnya masih sangat buruk. Khususnya untuk kata "dodi" dan "tadi". Banyak dari responden yang melakukan uji coba terhadap system ini tidak dapat menangkap atau mengerti kalimat percobaan yang diucapkan oleh system. Responden tersebut hanya dapat menangkap kalimat "368". Hal ini disebabkan karena suku kata yang terbentuk oleh kalimat "368" ini masih merupakan suku kata asli yang didapat dari *file* rekaman sumber data. Sedang suku kata lainnya sebagian besar sudah merupakan campuran dari kata-kata yang berbeda. Tabel 5.2 berikut menunjukkan hasil dari kuesioner yang

diberikan kepada beberapa responden yang sebagian besar adalah mahasiswa elektro UI angkatan 2005.

Tabel 5.3. Hasil kuesioner.

Resp	Alami	Jelas	Intonasi	Puas	Kata yang jelas		Kalimat
1	3	2	5	2	368		368
2	3	2	4	5	jam	delapan	kijang taun kijang betina 360 jam 8
3	1	1	4	1	968		...panjang... 968
4	3	3	4	3	368		...panjang... 368
5	3	2	2	4			...didengar tiga ratus...jam 8
6	4	2	4	2	368		...368
7	5	2	3	3	368		yang dengar 368
8	4	2	4	3	8	didengar	yang paling jelas yang paling didengar
							368
9	4	3	4	3	368		penelitian tahun 368
10	3	2	3	2	8		8
11	3	2	5	3	368		panjang tali yang di tengah 368
12	4	2	3	2	368		368
13	4	3	4	3	368		tarif yang didengar 368
14	4	3	4	4	dengar	368	...yang didengar 368
15	4	2	4	4	368		kopi tiam kopi tiam...368
16	3	2	4	2	968		kijang jantan kijang betina 968

Dari hasil kuesioner di Tabel 5.3 dapat dilihat bahwa tingkat kealamian (*naturalness)* untuk system TTS yang dibuat sudah baik. Dari 10 kata, hanya 5 kata saja yang dapat didengar dengan baik. Jadi sekitar 50% yang dapat dipahami dari kalimat ini. Hanya untuk kejelasan (*intelligibility)* sangat buruk. Namun, bila dianalisis pada *spectrogram* dari kalimat percobaan (dijelaskan pada sub-bab 5.5), sebenarnya tingkat kealamian yang terjadi masih buruk dalam hal durasi tiap suku kata yang tidak tepat, *pitch* yang tiba-tiba mengalami kenaikan atau penurunan. Para responden memberi nilai yang baik pada kategori tingkat kealamian sebab, presuposisi mereka tentang kealamian adalah bahwa suara ucapan yang didengar tidak seperti robot dalam hal intonasinya saja, tidak dalam hal durasi tiap suku kata. Penulis juga sebelumnya memiliki presuposisi yang sama tentang pengertian dari tingkat kealamian ini.

Yang menjadi penyebab utama dari *error* ini adalah karena kualitas *file* rekaman sumber yang tidak baik. File rekaman tersebut merupakan suatu rekaman khotbah di gereja. Karena proses perekaman yang dilakukan pada waktu itu tidak pada ruangan yang bebas *noise*, maka *file* rekaman yang tercipta juga mengandung banyak noise yang mengurangi kualitas suara rekaman. Dampak dari noise ini mengganggu penulis ketika melakukan proses pemotongan (*trimming*)

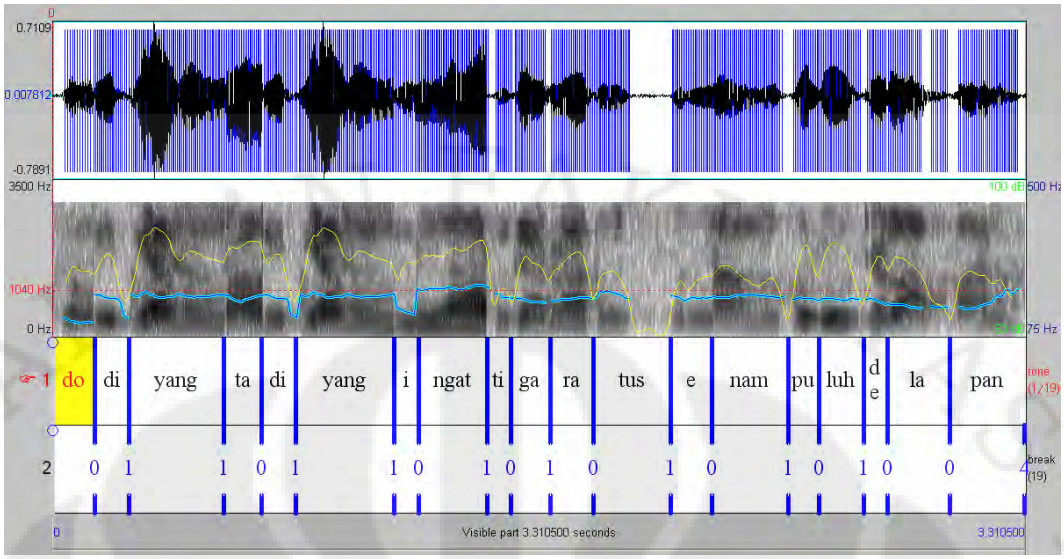
file suara untuk mendapatkan suku kata dengan *timing* yang tepat. Proses pemotongan *file* yang tidak tepat menghasilkan ucapan yang terdengar tidak jelas bahkan aneh ketika *file* suku kata tersebut digabungkan dengan suku kata yang lain. Ditambah lagi *noise* yang masih terkandung di dalam tiap *file* suku kata semakin memperburuk intellijibilitas ucapan.

Selain factor yang disebutkan di atas, factor lainnya adalah karena ketidakcocokan file suatu suku kata yang digabungkan dengan file suku kata yang lain. Ketidakcocokan tersebut berupa *pitch* yang tidak seimbang antar suku kata, seperti pada kata "dodi" di kalimat percobaan 1. Kata "do" memiliki *pitch* yang terlalu rendah dibandingkan dengan kata "di", sehingga kata "do" dan "di" ketika digabungkan terasa tidak cocok kedengarannya dan kata "do" itu sendiri hampir tidak kedengaran karena didominasi oleh suku kata yang memiliki pitch tinggi.

Meskipun segi intellijibilitas tidak baik, namun dari segi kealamian (*naturalness*), dapat dikatakan cukup baik. Intonasi yang terdengar pada kalimat percobaan tidaklah *monotone*. Hal ini dikarenakan ucapan pada rekaman memiliki intonasi yang tidak *monotone*. Sehingga suku kata yang dihasilkan juga memiliki variasi prosodi.

5.5. Analisis *signal* ucapan berikut dengan atributnya.

Gelombang ucapan gabungan yang dihasilkan dapat dilihat pada Gambar 5.3. di bawah ini. Gelombang ucapan yang ditunjukkan sudah berikut dengan *spectrogram*, *pitch tracking*, dan *break*. Gambar ini diperoleh dengan menggunakan Praat.



Gambar 5.3. Gelombang ucapan pada kalimat percobaan 1 beserta dengan atributnya.

Signal ucapan gabungan tersebut tidak diperoleh secara langsung dari program, melainkan dengan menggunakan software WavePad Editor. Tiap *file* suku kata di-*load* sehingga tergabung membentuk kalimat percobaan 1. Hasil ucapan yang dihasilkan agak berbeda dengan ucapan yang dikeluarkan oleh program. Pada program, durasi ucapan yang dihasilkan berdurasi sekitar 4.09 detik, sedang durasi ucapan yang diperoleh dengan software WavePad adalah 3.299 detik. Gekombang ucapan yang baru ini tidak terdapat jeda karena proses me-loading *file wav* seperti yang terjadi pada program TTS. Oleh karena itu, durasi yang dimiliki oleh gelombang ucapan yang baru lebih pendek.

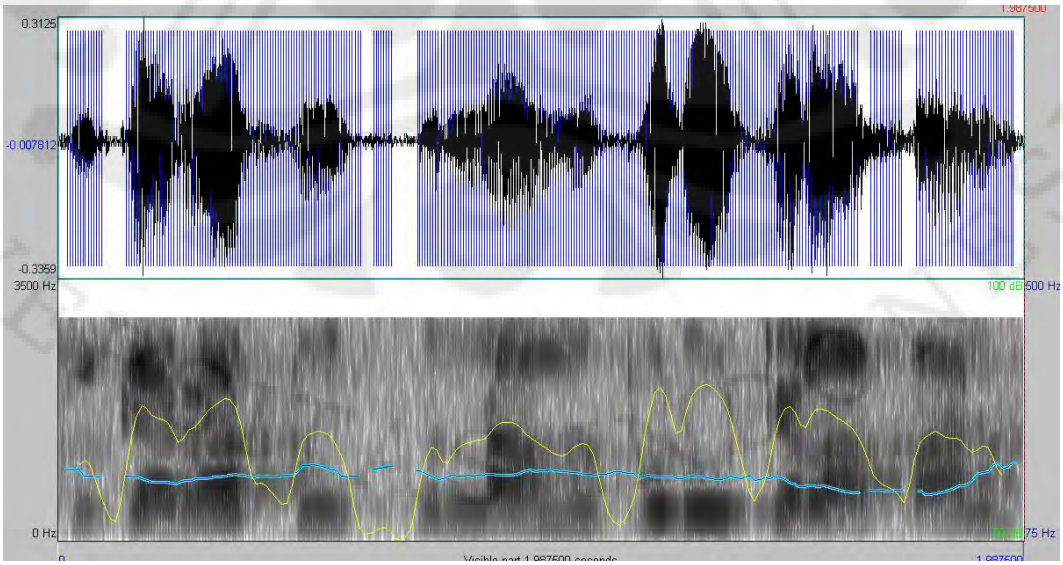
Pada Gambar 5.3 tersebut dapat dilihat segmen-segmen suku kata beserta dengan *break* antar suku kata yang disesuaikan dengan yang ada di *database*. Penulis tidak fokus pada ketepatan dalam menentukan ToBI dari kalimat ini. Jadi, *break* dan *accent* yang telah ditetapkan terlebih dahulu sangat mungkin mengandung kesalahan dalam penentuan ToBI setelah mengalami penggabungan. Kesalahan tersebut khususnya pada terletak penentuan *tone.*

Setting-an *spectrogram* yang digunakan di dalam Praat adalah sebagai berikut:

1. Fungsi *window* yang digunakan adalah *Hanning window*
2. *Range* frekuensi : 0 – 3500 Hz
3. Durasi *window* : 5 ms

4. *Dynamic range* : 70 dB
5. *Magnitude* maksimum : 60 dB/Hz
6. *Setting* yang lain adalah *default*

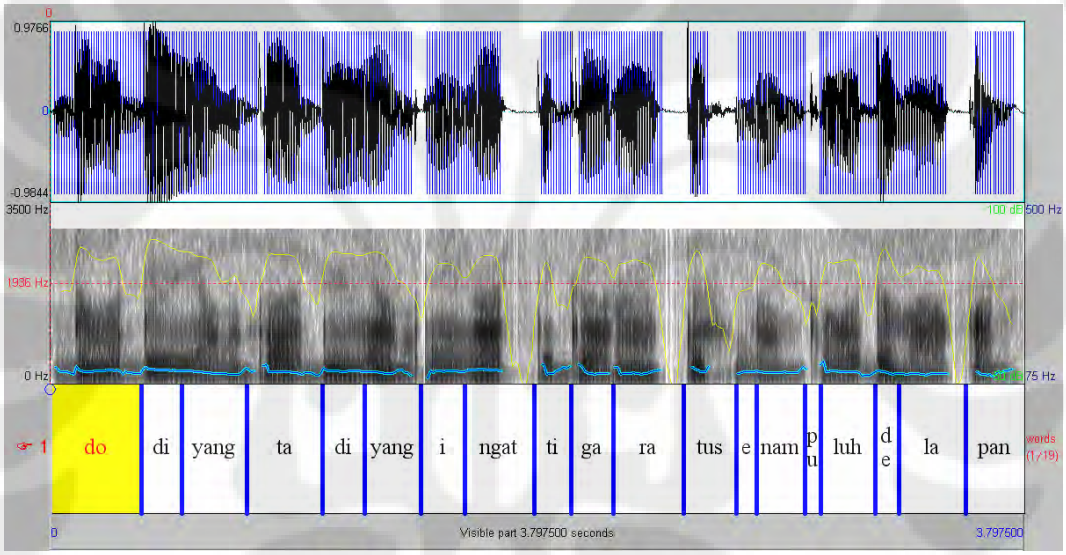
Pada *spectrogram* yang dihasilkan, dapat dilihat dengan cukup jelas untuk beberapa suku kata adanya ketidaksinkronan antara *spectrogram* suku kata yang satu dengan *spectrogram* suku kata yang lain. Hal ini menandakan bahwa barisan suku kata yang dipilih kurang tepat. Bila dianalisis pada *spectrogram*-nya, hal tersebut memiliki dampak pada pengurangan tingkat kealamian, meskipun secara pendengaran akan terdengar cukup alami dalam hal perbedaan intonasi yang cukup jelas. Namun tidak dalam hal durasi. Ketidakalamian tersebut meliputi dalam hal durasi tiap suku kata yang tidak tepat, *pitch* yang secara mendadak mengalami kenaikan atau penurunan. Ketidakalamian ini paling jelas terjadi pada kata "dodi", "tadi", dan "ingat". Untuk kata 368, pada *spectrogram* tidak terlalu terlihat adanya ketidaksesuaian. *Spectrogram* yang terbentuk pada kata 368 terlihat lebih alami dibandingkan dengan kata-kata yang lain. Kejadian tersebut dikarenakan pada kata 368, suku kata-suku kata yang digunakan masih merupakan suku kata yang asli dari kata 368 tersebut karena program tidak dapat memilih suku kata yang lain disebabkan keterbatasan *database*. *Spectrogram* kata 368 itu dapat dibandingkan dengan sumber data yang aslinya seperti yang terlihat pada Gambar 5.4 berikut ini.



Gambar 5.4. *Spectrogram* dari kata 368 yang asli.

Ketidakalamian tersebut merupakan akibat dari *database* yang dimiliki sangat terbatas dan berkualitas rendah. *Database* yang digunakan pada system ini memiliki banyak keterbatasan baik dalam hal kualitas maupun kuantitas. Kuantitas yang sedikit membatasi program dalam memilih suku kata dengan syarat tertentu. Kualitas yang rendah membuat program mengeluarkan ucapan dengan durasi yang tidak tepat dan intellijibilitas yang rendah. Alhasil, baik intellijibilitas dan tingkat kealamian akan buruk. Faktor ini merupakan faktor dominan yang mengurangi intellijibilitas dan tingkat kealamian system.

Pada Gambar 5.5 di bawah ini ditunjukkan *spectrogram* hasil dari rekaman ucapan penulis mengucapkan kalimat yang sama.



Gambar 5.5. *Spectrogram* dari kalimat percobaan yang diucapkan oleh penulis.

Setting-an *spectrogram* yang digunakan masih sama seperti sebelumnya. Namun, ketika menentukan suku kata, setting-an tersebut penulis ubah-ubah untuk memudahkan dalam menentukan suku kata.

Dari *spectrogram* pada Gambar 5.5, dapat dilihat bahwa antara suku kata yang satu dengan suku kata yang lain tampak alami, tidak terdapat perbedaan yang sangat signifikan antara *spectrogram* suku kata yang satu dengan *spectrogram* suku kata yang lain seperti yang terjadi pada Gambar 5.3. *Pitch* tracking yang terjadi tentu saja berbeda sebab *pitch* yang dimiliki penulis dengan *pitch* yang dimiliki oleh pak Eka Darmaputera adalah berbeda. *Pitch* yang terjadi

pada Gambar 5.5 memang tidak mengalami kenaikan atau penurunan yang mendadak seperti pada Gambar 5.3. Hal ini tentu membuat ucapan yang dihasilkan lebih alami. Namun intonasi yang dimiliki oleh Gambar 5.5, menurut penilaian penulis, tidak lebih baik bila dibandingkan dengan intonasi yang dimiliki oleh Gambar 5.3. Intonasi yang dimiliki Gambar 5.5 memiliki karakteristik yang datar-datar saja (*monotone*) sedang intonasi yang dimiliki Gambar 5.3 bervariasi. Sehingga ucapan yang dihasilkan pada Gambar 5.3 lebih "enak" didengar dan terkesan tidak membosankan. Lain halnya dengan Gambar 5.5, bila berdurasi cukup lama, maka akan membuat pendengar merasa jenuh.

Kelebihan dari ucapan pada Gambar 5.5 adalah dalam hal durasi suku kata yang dihasilkan. Durasi yang dimilikinya memiliki ketepatan yang lebih baik dibandingkan dengan durasi yang dimiliki oleh Gambar 5.3. Ada 2 faktor mengapa durasi yang dimiliki oleh system tidak baik.

1. Keterbatasan *database*.

   Database system tidak memiliki data ucapan suku kata yang cukup sehingga suku kata dengan durasi tertentu yang cocok berpasangan dengan suku kata tertentu yang lain malahan berpasangan dengan suku kata yang tidak tepat. Hasilnya adalah pada kata hasil bentukan suku kata tersebut dapat memiliki ketidaksesuaian durasi. Suku kata awal dapat memiliki durasi yang terlalu singkat dan suku kata berikutnya memiliki durasi yang lebih panjang. Hal ini tentu mengurangi tingkat kealamian, sebab akan terdengar aneh bila seseorang mengucapkan suatu kata dengan cara mengucapkan suku kata yang pertama dengan begitu singkat, sedang suku kata yang lain diucapkan dengan berdurasi lebih panjang.

2. Durasi yang tidak diperhitungkan pada modul Prosody Generator.

   Dalam system TTS yang dibuat, durasi yang diperlukan oleh suku kata tidak penulis perhitungkan di dalam modul ProsodyPredicting. Hal ini tentu saja sangat mempengaruhi system sebab sistem tidak mengetahui durasi yang diperlukan agar dapat menghasilkan ucapan yang baik. Untuk dapat menentukan durasi yang tepat untuk tiap unit suku kata memang sangat sulit. Sebab banyak faktor yang harus diperhitungkan untuk dapat menentukan durasi, seperti jumlah suku kata di dalam suatu kalimat, letak suku kata

tersebut di dalam suatu kalimat, intensitas suku kata, tanda baca yang mengikuti suku kata, konteks kalimat, dan faktor lainnya yang belum penulis ketahui. Namun, penulis mengajukan ide untuk menentukan durasi ini. Ide penulis adalah dengan menghitung jumlah suku kata yang ada di dalam suatu kalimat. Kemudian merata-ratakan nilai durasi suku kata dengan cara mengambil nilai rata-rata dari nilai rata-rata suku kata yang bersangkutan yang ada di dalam *database*. Nilai durasi rata-rata yang dihasilkan adalah nilai durasi rata-rata yang diperlukan oleh tiap suku kata di dalam kalimat tersebut. Durasi tiap suku kata paling tidak harus memiliki nilai yang berkisar dari nilai durasi rata-rata tersebut. Metode ini belum mempertimbangkan faktor yang lainnya seperti yang telah disebutkan sebelumnya. Namun, paling tidak metode ini dapat dicoba untuk dilihat hasilnya.

Tabel 5.4 berikut ini menunjukkan nilai dari perhitungan MFCC dan nilai *Manhattan Distance* untuk MFCC pada kalimat percobaan yang diucapkan oleh penulis:

Tabel 5.4. Data MFCC untuk kalimat percobaan 1 yang diucapkan oleh penulis.

Kata	Suku Kata		MFCC						Manhattan Distance
			1	2	3	4	5	6	
dodi	1	do1	42.68	44.18	44.11	43.04	59.86	60.11	50.15
	2	di1	59.14	60.4	57.62	55.93	55.78	55.26	
yang		yang1	55.64	56.36	58.87	58.27	58.02	54.97	
tadi	1	ta1	41.37	56.59	46.73	61.14	60.41	61.04	0.52
	2	di2	52.33	55.72	55.76	54.92	53.89	54.14	
yang		yang2	58.42	59.19	57.98	56.09	52.65	43.24	
ingat	1	i1	49.08	54.2	53.17	53.21	52.02	50.84	28.81
	2	ngat1	50.84	55.58	58.73	58.85	59.2	58.13	
tiga	1	ti1	51.62	55.62	52.19	38.9	38	37.63	66.48
	2	ga1	53.94	56.03	57.8	58.46	57.23	56.98	
ratus	1	ra1	56.91	56.46	56.83	57.16	56.21	55.23	18.86
	2	tus1	55.47	60.85	59.43	52.51	46.72	44.96	
enam	1	e1	46.18	45.3	0	0	0	0	213.61
	2	nam1	57.27	55.39	53.76	51.99	44.47	42.21	
puluh	1	pu1	44.96	46.96	0	0	0	0	243.41
	2	luh1	56.14	54.08	55.31	55.96	57.04	56.8	
delapan	1	de1	61.03	60.73	56.45	0	0	0	145.1
	2	la1	49.41	52.11	54.81	56.92	55.89	54.17	18.42
	3	pan1	47.49	59.93	56.91	53.94	45.69	40.93	

Dan Tabel 5.5 berikut ini menunjukkan nilai MFCC dan Manhattan Distance-nya untuk kalimat percobaan 1 yang diucapkan oleh sistem:

Tabel 5.5. Data MFCC untuk kalimat percobaan 1 yang diucapkan oleh sistem.

Kata	Suku Kata		MFCC						Manhattan Distance
			1	2	3	4	5	6	
dodi	1	do1	38.49	47.34	50.11	48.82	48.82	0	28.97
	2	di1	48.24	47.43	47.5	41.56	39.56	0	
yang		yang1	50.76	57.93	60.45	63.62	63.45	60.42	
tadi	1	ta3	56.91	57.5	58.5	59.57	58.4	0	66.59
	2	di1	48.24	47.43	47.5	41.56	39.56	0	
yang		yang1	50.76	57.93	60.45	63.62	63.45	60.42	
ingat	1	i1	44.72	44.81	46.18	0	0	0	188.22
	2	ngat1	50.85	49.9	50.62	58.85	55.68	58.03	
tiga	1	ti2	40.26	45.82	44.89	0	0	0	87.78
	2	ga2	55.09	55.41	54.38	53.87	0	0	
ratus	1	ra1	52.04	54.93	56.65	57.55	57.56	50.3	63.52
	2	tus1	48.4	45.67	46.66	46.31	40.96	37.51	
enam	1	e1	45.89	0	0	0	0	0	244.32
	2	nam1	43.4	43.63	43.9	45.43	53.64	55.23	
puluh	1	pu1	48.4	49.84	45.47	0	0	0	135.76
	2	luh1	48.49	49.17	47.02	46.6	46.17	40.68	
delapan	1	de1	43.93	53.93	57.47	0	0	0	174.82
	2	la2	54.77	55.81	54.78	55.01	54.14	50.26	24.65
	3	pan1	42.98	51.39	52.57	53.27	53.4	54.01	

Dari data-data pada table 5.4 dan 5.5 di atas, dapat dilakukan perbandingan nilai dari Manhattan Distance untuk MFCC koefisien ke 0. Namun, analisis yang dapat dilakukan belum dapat dilakukan secara mendalam Karena penentuan nilai c0 tersebut dilakukan dengan menggunakan durasi *window* dan *time step* yang cukup panjang yang ditentukan pada software Praat. Sehingga sangat mungkin untuk mengandung *error* dari data tersebut yang berdampak pada tidak dapat direpresentasikannya *signal* ucapan dengan baik. Jadi, untuk kasus ini, nilai *Manhattan Distance* yang besar untuk MFCC ini belum tentu melambangkan bahwa kata yang dihasilkan adalah buruk. Demikian juga sebaliknya, nilai *Manhattan distance* yang kecil belum tentu melambangkan kata yang dihasilkan adalah baik. Seperti kata "dodi", misalnya. Nilai *Manhattan Distance* untuk kata ini pada ucapan yang dihasilkan oleh penulis bernilai lebih tinggi dibandingkan

yang diucapkan oleh sistem. Namun yang diucapkan oleh penulis memiliki kejelasan yang lebih baik dibandingkan yang diucapkan oleh sistem. Sehingga nilai *Manhattan distance* tidak menjelaskan tingkat kejelasan (intellijibilitas) secara keseluruhan.

Seperti yang dijelaskan sebelumnya, bahwa nilai c0 dari MFCC melambangkan tingkat kekerasan suatu *signal* suara. Nilai *Manhattan distance* dari c0 ini berarti dapat melambangkan perbedaan tingkat kekerasan antara suku kata yang satu dengan suku kata berikutnya. Sehingga nilai *Manhattan distance* yang besar melambangkan tingkat kekerasan antara suku kata yang satu dengan suku kata berikutnya adalah besar. Bila bernilai kecil berarti perbedaan kekerasannya adalah kecil. Tapi hal ini belum bisa menjadi tolok ukur sepenuhnya. Sebab, juga ada ketergantungan dengan suku kata yang diucapkan. Ada perbedaan daya *signal* yang dihasilkan dari *signal* ucapan *voiced* dan *unvoiced*. *Signal* ucapan *voiced,* pada umumnya memiliki energi yang lebih besar dibandingkan dengan *unvoiced.*

BAB VI
KESIMPULAN

1. Sistem TTS memiliki 2 modul utama, yaitu NLP dan DSP.
2. Pembuatan *database* ucapan dimana unit berupa suku kata yang diambil dari rekaman ucapan yang berupa khotbah atau pidato atau yang sejenisnya dapat meningkatkan tingkat kealamian ucapan yang dihasilkan dalam hal intonasi.
3. Proses pemotongan *file wav* dengan *timing* yang tepat diperlukan untuk menghasilkan ucapan dengan intellijibilitas yang baik.
4. Untuk mengurangi jeda yang tidak diperlukan antar suku kata, diperlukan teknik pemrograman yang baik dan efisien.
5. Sistem TTS yang dibuat pada skripsi ini masih banyak kekurangan baik dalam hal intellijibilitas dan tingkat kealamian dikarenakan *database* yang sangat sedikit.
6. Program TTS yang dibuat masih kurang efisien karena waktu yang diperlukan untuk memulai ucapan masih cukup lama dan antar suku kata terdapat jeda walau tidak diberi jeda.
7. Sistem TTS dengan metode *Unit Selection Synthesis* membutuhkan *database* yang sangat besar untuk dapat menghasilkan ucapan yang bermutu.

DAFTAR ACUAN

1. "Arman, Arry Akhmad", "**Introduction to Text to Speech Technology and Applications",** Diunduh dari: http://www.slideshare.net/kupalima/introduction-to-teks-to-speech-technology-and-applications-presentation, tanggal 2 Januari 2009.
2. "Zitzewitz, Paul W.; Elliot, Todd George; Haase, David G.; Harper, Kathleen A.; Herzog, Michael R.; Nelson, Jane Bray; Nelson, Jim; Schuler, Charles A.; Zorn, Margaret K.", "**Physics, Principles & Problems**", Columbus: McGraw-Hill, 2005.
3. "Huang, Xuedong; Acero, Alex; Hon, Hsiao-Wuen", "**Spoken Language Processing**", New Jersey: Prentice-Hall, Inc. 2001.
4. "Arman, Arry Akhmad", "**Koversi dari Text ke Ucapan"**, Diunduh dari: http://indotts.melsa.net.id/knowledge.html, tanggal 30 November 2008.
5. "Goldberg, Randy & Riek, Lance", "**A Practical Handbook of Speech Coders**", Boca Raton: CRC Press LLC, 2000.
6. "Taylor, Paul". "**Text-to-Speech Synthesis**". United Kingdom: Cambridge University Press, 2007.
7. "Dutoit, T", "**A Short Introduction to Text-to-Speech Synthesis**", Diunduh dari: http://www.coli.uni-saarland.de/~dominika/Dutoit-eee97.pdf , tanggal 2 Januari 2009.
8. "Muslich, Masnur" "**Fonologi Bahasa Indonesia, Tinjauan Deskriptif Sistem Bunyi Bahasa Indonesia**", Jakarta: PT. Bumi Aksara, 2008.
9. "Benesty, Jacob; Sondhi, M. Mohan; Huang, Yiteng", "**Springer Handbook of Speech Processing"**, Berlin Heidelberg: Springer-Verlag, 2008.
10. http://en.wikipedia.org/wiki/*Formant*
11. "Mertins, Alfred", "***Signal* Analysis"**, England: John Wiley & Sons Ltd, 1999.
12. http://en.wikipedia.org/wiki/Mel_scale

13. “Xi Zhou”, “YunFu”, “Ming Liu”, “Johnson, Mark Hasegawa”, “Huang, Thomas S.”, “Robust Analysis And Weighting on MFCC Components for Speech Recognition And Speaker Identification”
14. http://ocw.mit.edu
Situs untuk mengunduh mata kuliah ToBI yang diunduh pada bulan Oktober tahun 2009.

DAFTAR PUSTAKA

"Abolrous, Sam A.", "**Learn Pascal**". Texas: Wordware Publishing, Inc., 2000.

"Arman, Arry Akhmad", "**Introduction to Text to Speech Technology and Applications"**. Diunduh dari: http://www.slideshare.net/kupalima/introduction-to-teks-to-speech-technology-and-applications-presentation, tanggal 2 Januari 2009.

"Arman, Arry Akhmad". "**Karakteristik Sinyal Ucapan",** Diunduh dari http://indotts.melsa.net.id/knowledge.html. tanggal 30 November 2008.

"Arman, Arry Akhmad", "**Koversi dari Text ke Ucapan"**, Diunduh dari: http://indotts.melsa.net.id/knowledge.html. tanggal 30 November 2008.

"Benesty, Jacob; Sondhi, M. Mohan; Huang, Yiteng", "**Springer Handbook of Speech Processing"**. Berlin Heidelberg: Springer-Verlag, 2008.

"Black, Alan W. & Lenzo, Kevin A.", "**Building Synthetic Voices**". Diunduh dari: http://festvox.org/bsv/bsv.ps.gz. tanggal 14 Januari 2009.

"Bucknall, Julian", "**The Tomes of Delphi Algorithms And Data Structures**". Texas: Wordware Publishingn Inc., 2001.

"Canto, Marco", "**Mastering Delphi 7**". Sybex Inc., Alameda, CA, 2003.

"Chaer, Abdul", "**Leksikologi & Leksikografi Indonesia**". Jakarta: PT. Rineka Cipta, 2007.

"Ding-Zhu Du, Ker-I Ko", "**Problem Solving in Automata, Languages, and Complexity**". New York: John Wiley & Sons, Inc., 2001.

"Dutoit, T", "**A Short Introduction to Text-to-Speech Synthesis**", Diunduh dari: http://www.coli.uni-saarland.de/~dominika/Dutoit-eee97.pdf. tanggal 2 Januari 2009.

"EL-IMAM, YOUSIF A. & ZURAIDA MOHAMMED DON", "**Text-to-Speech of Standard Malay**", INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 3. Pp. 129-146, 2000.

"Goldberg, Randy & Riek, Lance", "**A Practical Handbook of Speech Coders**". Boca Raton: CRC Press LLC, 2000.

"Hanov, Steve", "**Automatic Letter-to-Sound Rules for Speech Synthesis**". 2007.

"Hariyanto, Bambang", "**Teori Bahasa, Otomata, dan Komputasi serta Terapannya**". Bandung: Penerbit Informatika, 2004.

"Huang, Xuedong; Acero, Alex; Hon, Hsiao-Wuen", "**Spoken Language Processing**". New Jersey: Prentice-Hall, Inc. 2001.

"Hunt, Andrew J. & Black, Alan W.", "**Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database**". Proc. IEEE-ICASSP'96, pp.373-376, 1996.

"Jurafsky, Daniel; Martin, James H.", "**Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**". New Jersey: Prentice-Hall, 1999.

"Kadir, Abdul", "**Dasar Aplikasi Database MySQL Delphi** (Edisi II)". Yogyakarta: Penerbit Andi, 2003.

"Munir, Rinaldi", "**Matematika Diskrit** (Edisi III)", Bandung: Penerbit Informatika, 2005.

"Muslich, Masnur", "**Fonologi Bahasa Indonesia, Tinjauan Deskriptif Sistem Bunyi Bahasa Indonesia**". Jakarta: PT. Bumi Aksara, 2008.

"Rabiner, Lawrence R. & Schafer, Ronald W.","**Introduction to Digital Speech Processing**". Foundations and Trends in *Signal* Processing Vol 1, 2007.

"Raharjo, Budi", "**Teknik Pemrograman Pascal (**Edisi II)". Bandung: Informatika Bandung, 2008.

"Saiful Bahri, Kusnassriyanto & Sjachriyanto, Wawan", "**Teknik Pemrograman Delphi** (Edisi II)". Bandung: Informatika Bandung, 2008.

"Sef, Toma'z & Mtja'z Gams", "**SPEAKER (GOVOREC): A Complete Slovenian Text-to Speech System**". INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 6, pp. 277–287**,** 2003.

"Sudargo, Paulus", "**Pemrograman Berorientasi Objek Menggunakan Delphi"**. Yoyakarta: Penerbit Andi, 2004.

"Tatham, Mark & Morton, Katherine", "**Development in Speech Synthesis**". England: John Wiley & Sons Ltd, 2005.

"Taylor, Paul". "**Text-to-Speech Synthesis**". United Kingdom: Cambridge University Press, 2007.

"Tri Putra A. N., Dedi". "**Cara Mudah Belajar Delphi**". Semarang: Neomedia Press, 2006.

"Zitzewitz, Paul W.; Elliot, Todd George; Haase, David G.; Harper, Kathleen A.; Herzog, Michael R.; Nelson, Jane Bray; Nelson, Jim; Schuler, Charles A.; Zorn, Margaret K.", "**Physics, Principles & Problems**". Columbus: McGraw-Hill, 2005.

Situs Internet:

http://en.wikipedia.org/wiki/*Formant*

http://en.wikipedia.org/wiki/MBROLA

http://indotts.melsa.net.id

http://www.ling.su.se/staff/hartmut/kempln.htm

http://pebbie.wordpress.com/ diakses pada tanggal 2 Mei 2009

http://www.speech.cs.cmu.edu/11-492/slides/

```
procedure TTools.SearchCoordinate(SylTotal, syl1, syl2, syl3, n: integer;
  var x, y, z: integer);
var
  remainder: integer;
begin
  case SylTotal of
  2:
    begin
      if n> syl2 then
      begin
        if (n mod syl2=0) then
        begin
          x:= n div syl2;
          y:= n - ((x-1)*syl2);
          z:= 0;
        end
        else
        begin
          x:= (n div syl2) + 1;
          y:= n - ((x-1)*syl2);
          z:= 0;
        end;
      end
      else
      begin
        x:= 1;
        y:= n;
        z:= 0;
      end;
    end;
```

```
 3:
   begin
     if n<= syl3 then
     begin
       x:= 1;
       y:= 1;
       z:= n;
     end
     else if ((n<= syl2*syl3)and(n>syl3)) then
     begin
       x:= 1;
       if ((n mod syl3) = 0) then
       begin
         y:= n div syl3;
         z:= n - ((y-1)*syl3);
       end
       else
       begin
         y:= (n div syl3) + 1;
         z:= n - ((y-1)*syl3) ;
       end;
     end
     else if n> (syl2*syl3) then
     begin
       x:= n div(syl2*syl3) + 1;
       remainder:= n - ((x-1)*syl2*syl3);
       y:= (remainder div syl3) + 1;
       z:= remainder - ((y-1)*syl3);
     end;
   end;
 end;
end;
```