IMPLEMENTASI *LIBRARY* POCKETSPHINX UNTUK PENGENALAN *VOICE COMMAND* BERBAHASA INDONESIA SECARA *OFFLINE*

SKRIPSI

LABORATORIUM SISTEM KOMPUTER DAN ROBOTIKA

Diajukan untuk memenuhi sebagian persyaratan
mencapai gelar Sarjana Komputer

Disusun Oleh :

MOKHAMMAD HILMAN FATAH

NIM. 105060807111093

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

MALANG

2015

LEMBAR PERSETUJUAN

IMPLEMENTASI *LIBRARY* POCKETSPHINX UNTUK PENGENALAN *VOICE COMMAND* BERBAHASA INDONESIA SECARA *OFFLINE*

SKRIPSI

LABORATORIUM SISTEM KOMPUTER DAN ROBOTIKA

Diajukan untuk memenuhi sebagian persyaratan
mencapai gelar Sarjana Komputer

Disusun oleh :

MOKHAMMAD HILMAN FATAH

NIM. 105060807111093

Skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 30 Januari 2015

Pembimbing I	Pembimbing II
Eko Setiawan, S.T., M.T.	**Budi Darma Setiawan, S.Kom., M.Cs.**
NIK. 870610 06 1 1 0256	**NIP. 198410152014041002**

LEMBAR PENGESAHAN

IMPLEMENTASI *LIBRARY* POCKETSPHINX UNTUK PENGENALAN *VOICE COMMAND* BERBAHASA INDONESIA SECARA *OFFLINE*

SKRIPSI

LABORATORIUM SISTEM KOMPUTER DAN ROBOTIKA

Diajukan untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :
MOKHAMMAD HILMAN FATAH
NIM. 105060807111093

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 16 Januari 2015

Penguji I	Penguji II
Wijaya Kurniawan, S.T., M.T.	**Barlian Henryranu Prasetio, S.T., M.T.**
NIK. 820125 16 1 1 0418	**NIK. 82102406110254**

Penguji III

Gembong Edhi Setyawan, S.T., M.T.
NIK. 76120116110373

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Mardji., M.T.
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku yaitu UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70.

Undang-undang No. 20 Tahun 2003 Pasal 25 ayat 2 yang berisi "Lulusan perguruan tinggi yang karya ilmiahnya digunakan untuk memperoleh gelar akademik, profesi, atau vokasi terbukti merupakan jiplakan dicabut gelarnya." dan Pasal 70 yang berisi "Lulusan yang karya ilmiah yang digunakannya untuk mendapatkan gelar akademik, profesi, atau vokasi sebagaimana dimaksud dalam Pasal 25 ayat (2) terbukti merupakan jiplakan dipidana dengan pidana penjara paling lama dua tahun dan/atau pidana denda paling banyak Rp 200.000.000,00 (dua ratus juta rupiah)."

Malang, 30 Januari 2015

Mahasiswa,

Mokhammad Hilman Fatah

105060807111093

KATA PENGANTAR

Puji dan syukur saya ucapkan kepada Allah SWT karena atas rahmat dan hidayah-Nya saya dapat menyelesaikan Tugas Akhir ini sebagai salah satu persyaratan untuk menyelesaikan studi di Program Studi Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.

Saya menyadari bahwa Tugas Akhir ini dapat terselesaikan atas bantuan, petunjuk, dan bimbingan dari berbagai pihak yang telah membantu proses penyelesaiannya. Oleh karena itu, saya mengucapkan terima kasih banyak kepada :

1. Orang tua saya, Abdus Somad dan Taty Widarti yang tak henti-hentinya memberikan dorongan moril dan materil yang telah membantu hingga terselesaikanya skripsi ini.
2. Bapak Eko Setiawan, S.T., M.T. dan Bapak Budi Darma Setiawan, S.Kom, M.Cs. selaku pembimbing I dan pembimbing II yang telah memberikan bimbingan dan arahan dengan penuh kesabaran sehinga laporan skripsi ini dapat terselesaikan.
3. Mas Didit sebagai Laboran Sistem Komputer & Robotika Program Teknologi Informasi dan Ilmu Komputer yang membantu dalam pelaksanaan dan pengadaan perangkat untuk skripsi ini.
4. Bapak Drs. Mardji, M.T. dan Bapak Issa Arwani, S.T., M.T. selaku ketua dan sekretaris Progam Studi Informatika.
5. Bapak Barlian Henryranu Prasetio, S.T., M.T. selaku Kepala Laboratorium Sistem Komputer dan Robotika Sistem Komputer dan pengagas terciptanya topik skripsi ini.
6. Seluruh dosen pembina pada Progam Studi Informatika dan Ilmu Komputer yang telah membimbing penulis selama menjadi mahasiswa Universitas Brawijaya Malang.
7. Seluruh mahasiswa Program Teknologi Informasi dan Ilmu Komputer, khususnya teman – teman dan sahabat – sahabat saya (Arinda Hapsari Achnas, Fahmi, Tanty, Adam, Dendy, Icus, Alfa, Tamy, Adit, Bayu),

teman-teman kos kembang kertas serta teman-teman interface 2010 yang telah membantu terealisasinya skripsi ini.

8. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Penulis menyadari bahwa skripsi ini masih belum sempurna, maka penulis mengharapkan adanya kritik dan saran untuk kesempurnaan skripsi ini, serta Tugas Akhir ini dapat memberikan manfaat bagi pembaca sekaligus dapat menjadi bahan acuan untuk penelitian selanjutnya.

Malang, 30 Januari 2015

Penulis

ABSTRAK

Mokhammad Hilman Fatah 2014. : Implementasi *Library* PocketSphinx Untuk Pengenalan *Voice Command* Berbahasa Indonesia Secara *Offline*. Skripsi Program Studi Informatika / Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Eko Setiawan, S.T., M.T. dan Budi Darma Setiawan, S.Kom, M.Cs.

Pembangunan teknologi *voice command* yang berbasis *online* sudah banyak dikembangkan di Indonesia tetapi kebutuhan akan infrastruktur internet masih dalam tahap perkembangan sehingga sisi fungsionalitas dari penerapan teknologi tersebut tidak digunakan secara optimal. Oleh karena itu, dibutuhkan pembangunan teknologi *voice command* yang dijalankan secara *offline* sehingga dapat diterapkan pada perangkat teknologi lainnya yang tidak membutuhkan koneksi internet.

Aplikasi Pengenalan *voice command* berbahasa Indonesia dirancang menggunakan *library* PocketSphinx dengan menggunakan metode *Hidden Markov Model* yang membantu tingkat keakurasian pada pengenalan suara bahasa Indonesia. Pembangunan aplikasi ini dijalankan secara *offline* dan dapat mendeteksi kosakata bahasa Indonesia yang diucapkan oleh pembicara secara langsung.

Setelah dilakukan pengujian, aplikasi menunjukkan hasil yang cukup baik yaitu dapat mengidentifikasi suara bahasa Indonesia dengan tingkat akurasi sebesar 94.88% berdasarkan 7 koresponden suara pada saat melakukan *training* dan untuk hasil percobaan program secara langsung mencapai akurasi sebesar 79% berdasarkan 5 koresponden suara. Penulis yakin aplikasi *voice command speech to text* ini dapat terus dikembangkan lebih baik lagi dengan menambah jumlah kosakata, variasi suara dan mengestimasi parameter-parameter dalam proses training suara yang sangat mempengaruhi hasil pengenalan suara.

Kata kunci : *Voice Command, Library,* PocketSphinx*, Offline, Online, Hidden Markov Model*.

ABSTRACT

Mokhammad Hilman Fatah 2014: Implementation of PocketSphinx Library For Recognition Indonesian Voice Command In Offline. Thesis: Study Program Informatics / Computer Science, Information Technology Program and Computer Science, University of Brawijaya. Supervisor: Eko Setiawan, S.T., M.T. and Budi Darma Setiawan, S.Kom, M.Cs.

Development of technology online-based voice command has been developed in Indonesia, but the Internet infrastructure is still under development in this country, so the functionality of the technology is not good enough to use. Therefore, there is a chance to build this technology can be running in offline, so that it can be applied to other technological devices that not require an internet connection.

Application of recogniton voice command with Indonesian language designed by using PocketSphinx library and Hidden Markov Model which helps level of accuracy in speech recognition with Indonesia language. Development of this application run offline and can detect Indonesian vocabulary spoken by the speaker directly.

After testing, the application shows good results that can identify sound Indonesian with an accuracy rate of 94.88% based on 7 correspondent voices during training and for the results of the experimental program directly reach an accuracy of 79% based on 5 correspondent voices. The author believes the application voice command with Indonesia language can be developed even better by adding the amount of vocabulary, sound variations and estimating the parameters of the voice training process that greatly affect the results of speech recognition.

Keywords: Voice Command, Library, PocketSphinx, Offline, Online, Hidden Markov Model.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR PERSAMAAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Teknologi *voice command* merupakan teknologi yang memungkinkan komputer untuk mengenali dan mengidentifikasi kosakata yang diucapkan oleh suara seseorang [JAM-05]. Saat ini, teknologi tersebut sangat marak dikembangkan secara *online* demi terciptanya komunikasi yang memudahkan antar manusia untuk menggunakan teknologi tersebut. Namun, penggunaan pengenalan suara memiliki dampak yang cukup riskan yakni, suara yang diucapkan oleh seseorang harus mendeteksi suara dengan baik dan tidak memiliki jeda waktu diantara proses tersebut. Oleh karena itu, dibutuhkan koneksi internet yang cukup cepat dan stabil agar dapat menggunakan fasilitas teknologi tersebut dengan baik.

Teknologi *voice command* yang berbasis *online* di Indonesia sudah banyak dikembangkan, namun kebutuhan akan infrastruktur internet masih dalam tahap perkembangan sehingga sisi fungsionalitas dari penerapan teknologi tersebut tidak cukup baik digunakan. Teknologi tersebut dapat dengan baik dikembangkan di Indonesia, apabila diterapkan secara *offline* sehingga nantinya dapat digunakan pada perangkat teknologi lainnya yang tidak membutuhkan koneksi internet. Pengenalan *voice command* membutuhkan komputasi matematika pada saat mengenali suara yang diucapkan oleh manusia, sehingga kebutuhan akan pengaksesan yang cepat dan langsung (*realtime*) sangat dibutuhkan pada pengembangan teknologi ini. Oleh karena itu, peneliti menginginkan teknologi ini dapat dijalankan secara *offline* sehingga tidak menghambat komputasi pada saat mengenali suara seseorang.

Perancangan teknologi *voice command* ini menggunakan *library* PocketSphinx dengan metode *Hidden Markov Model* (HMM) yang dapat memaksimalkan hasil pengenalan suara yang diucapkan oleh seseorang. *Library* PocketSphinx merupakan perangkat lunak *open-source* yang digunakan untuk mengenali suara seseorang lalu mengkonversikannya dalam bentuk teks. PocketSphinx memudahkan penggunanya untuk memodifikasi Bahasa yang akan digunakan pada pembangunan aplikasi *voice command*. Pengembangan teknologi

ini memiliki banyak manfaat, apabila diterapkan pada teknologi lainnya, seperti pembangunan smarthome berbasis *voice control* dengan menggunakan bahasa Indonesia dan alat bantu yang menggunakan bahasa Indonesia bagi kaum lansia dan disabilitas untuk membantu mereka berkomunikasi sehari-hari.

Berdasarkan uraian permasalahan yang telah dipaparkan di atas, dan dengan perkembangan teknologi yang ada serta ketersediaan perangkat keras untuk mengimplementasikan teknologi tersebut, diharapkan bisa menjadi solusi dari masalah yang tela dipaparkan pada paragraf sebelumnya. Oleh karena itu disini penulis mengangkat sebuah skripsi dengan judul **"Implementasi *Library* PocketSphinx Untuk Pengenalan *Voice Command* Berbahasa Indonesia Secara *Offline*"**. Aplikasi ini dibuat berdasarkan pengukuran dan analisa untuk mengetahui kelayakannya. Serta dapat memberikan acuan untuk penerapan dalam studi kasus sehingga fitur *voice command* Berbahasa Indonesia ini dapat diimplementasikan dengan baik.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan di atas, maka rumusan masalah dapat disusun sebagai berikut:

1. Bagaimana cara merancang aplikasi dengan fitur *voice command* berbahasa Indonesia berbasis *offline* untuk mengenali pola suara orang Indonesia.
2. Bagaimana cara mengatur parameter *library* PocketSphinx agar dapat mengenali suara bahasa Indonesia.
3. Bagaimana kinerja *library* PocketSphinx dalam mengenali suara bahasa Indonesia yang disimulasikan pada komputer.

1.3. Batasan Masalah

Berdasarkan latar belakang dan rumusan masalah yang telah dikemukakan, penelitian ini mempunyai batasan-batasan masalah sebagai berikut:

1. Aplikasi *voice command* berbahasa Indonesia ini merupakan aplikasi pengenalan suara berbasis *offline* yang disimulasikan pada komputer.
2. Aplikasi *voice command* hanya dapat mengenali suara manusia dengan menggunakan bahasa Indonesia.

3. PocketSphinx merupakan *library* yang digunakan untuk mengimplementasikan aplikasi *voice command* berbahasa Indonesia.
4. Fitur yang digunakan pada *library* PocketSphinx ini hanya dapat mengubah dari suara manusia menjadi text (*speech to text*).
5. Pada saat melakukan *testing* program, aplikasi ini hanya dapat mengenali 25 kosakata bahasa Indonesia.

1.4. Tujuan

Tujuan skripsi ini adalah merancang, membangun aplikasi "Pengenalan Suara Bahasa Indonesia Menggunakan *Voice Command Speech to Text yang Berbasis Offline*" yang nantinya dapat diimplementasikan ke dalam perancangan *smarthome* berbasis *offline voice control* maupun untuk membangun teknologi yang dapat membantu kaum lansia dan disabilitas dalam kehidupannya sehari-hari.

1.5. Manfaat

Manfaat yang diperoleh dari penelitian skripsi ini adalah sebagai berikut:

1. Bagi Penulis
   a. Dapat menambah pengetahuan dan wawasan, serta dapat mengaplikasikan dan mensosialisasikan teori yang telah diperoleh selama perkuliahan.
   b. Dapat mendapatkan pengalaman lebih dalam mengkaji teknologi yang dapat digunakan sebagai awal mula perancangan *smarthome* berbasis *offline voice control* sekaligus membantu perancangan teknologi yang dapat dimanfaatkan oleh kaum lansia dan disabilitas.
2. Bagi Pengguna
   a. Menjadi user yang ikut ambil andil dalam pengembangan teknologi ini kedepannya melalui saran-saran ataupun kritik yang disampaikan.
   b. Dapat digunakan oleh penyandang cacat dan kaum lansia sebagai alat dan sarana pembantu media komunikasi.

c. Dapat digunakan sebagai *smarthome* dengan *voice control* berbasis *offline*.

1.6. Sistematika Penulisan

Penyusunan skripsi ini menggunakan kerangka pembahasan yang tersusun sebagai berikut:

BAB I PENDAHULUAN

Bab ini merupakan dasar dari penyusunan skripsi ini yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II TINJUAN PUSTAKA DAN DASAR TEORI

Bab ini berisi kajian pustaka, referensi mengenai perangkat-perangkat yang digunakan dalam mendukung pembuatan aplikasi atau sumber-sumber yang berhubungan dengan permasalahan dalam skripsi yang meliputi : Aplikasi *voice command* menggunakan *library* PocketSphinx dan Hidden Markov Model.

BAB III METODOLOGI PENELITIAN DAN PERANCANGAN

Bab ini menjelaskan bagaimana metodologi penelitian, memberikan gambaran umum aplikasi, perancangan untuk melakukan *training* suara dan *testing* suara pada *library* PocketSphinx dengan menggunakan Bahasa Indonesia.

BAB IV IMPLEMENTASI

Bab ini menjelaskan implementasi untuk untuk melakukan *training* suara dan *testing* suara pada *library* PocketSphinx dengan menggunakan bahasa Indonesia.

BAB V PENGUJIAN DAN ANALISIS

Bab ini menjelaskan hasil pengujian aplikasi berdasarkan parameter-parameter yang telah ditetapkan dan kemudian dilakukan analisa terhadap performansi berdasarkan hasil pengujian *training* dan *testing* suara.

BAB VI PENUTUP

Bab ini menjelaskan kesimpulan dan saran yang dapat diambil dari pengujian dan analisis berdasarkan perumusan masalah yang dibuat untuk merancang aplikasi *voice command* berbahasa Indonesia dan disertai saran sebagai acuan untuk pengembangan selanjutnya.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini membahas kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi "Implementasi *Library* PocketSphinx Untuk Pengenalan *Voice Command* Berbahasa Indonesia Secara *Offline*".

2.1. Kajian Pustaka

Kajian pustaka membahas penelitian yang telah ada dan yang diusulkan. Pada penelitian ini kajian pustaka diambil dari beberapa penelitian yang relevan dan pernah dilakukan penelitiannya sebelumnya. Pada penelitian pertama yang berjudul "Perancangan Program Aplikasi Android *Speech to Text* Bahasa Indonesia Dan Bahasa Inggris Menggunakan Metode Hidden Markov Model", Penelitian ini membahas tentang aplikasi android untuk penggunaan kamus dengan menggunakan fitur *speech to text* untuk bahasa Indonesia dan bahasa Inggris. Hasil pengujian didapatkan bahwa aplikasi ini dapat mengidentifikasi suara penulis dengan tingkat akurasi sebesar 91.76% untuk versi bahasa Indonesia dan 83.05% untuk versi bahasa Inggris. Maka dapat disimpulkan bahwa perancangan ini dapat berguna memudahkan bagi pengguna untuk belajar mengucapkan kata-kata sehari-hari dalam bahasa Indonesia dan bahasa Inggris serta penggunaan metode *Hidden Markov* yang membantu dalam perancangan aplikasi ini [MVE-12].

Pada penelitian yang kedua berjudul "Natural Speaker-Independent Arabic Speech Recognition System Based on Hidden Markov Models Using Sphinx Tools". Penelitian tersebut mengembangkan pengenalan suara bahasa Arab dengan menggunakan *library* CMU Sphinx. Selain itu aplikasi ini menggunakan teknik 5 *state Hidden Markov Model* (HMM) dengan menggunakan tiga state untuk pemodelan akustik *tri-phone*. Model bahasa mengandung uni-gram, bi-gram, dan tri-gram. Hasil pengujian pengenalan suara bahasa Arab ini mendapatkan akurasi diatas 90% [MAM-10].

Pada penelitian yang ketiga berjudul "Automatic Speech Recognition: Human Computer Interface For Kinyarwanda Language" merupakan aplikasi

pengenalan suara otomatis bagi bahasa Kinyarwanda. Sistem ini diimplementasikan dengan menggunakan *toolkit* HMM HTK oleh berdasarkan pelatihan kata-kata membuat kosakata pada data pelatihan HMM. Sistem dilatih diuji pada data lain dari data pelatihan dan hasil menunjukkan bahwa 94.47% dari data yang diuji serta diakui. Sistem ini dikembangkan dapat digunakan oleh pengembang dan peneliti yang tertarik pada pengenalan suara untuk bahasa Kinyarwanda dan bahasa Afrika terkait lainnya. Temuan penelitian dapat digeneralisasi untuk memenuhi pebendaharaan kosakata yang lumayan banyak dan dapat dikembangkan secara berkelanjutan [JAM-05].

Berdasarkan penelitian-penelitian yang telah dipaparkan di paragraph sebelumnya, disimpulkan bahwa hasil pengenalan suara menggunakan *library PocketSphinx* sudah cukup baik digunakan dalam berbagai bahasa. Pada penelitian kali ini, peneliti menginginkan hasil akurasi pelatihan (*training*) suara bahasa Indonesia sehingga memiliki hasil akurasi yang lebih baik dibandingkan penelitian sebelumnya. Peneliti akan mengestimasi parameter-parameter yang akan digunakan pada saat *training* suara berupa parameter *nfft*, *nfilt*, *upperf*, *lowerf* dan pada saat melakukan *testing* berdasarkan koresponden *training*, memiliki akurasi yang lebih baik dibandingkan penelitian sebelumnya. Pada penelitian kali ini juga akan menyesuaikan kosakata yang akan digunakan sebagai *smarthome system* berbasis *voice control* dan alat pembantu untuk kaum disabilitas dan lansia serta akan diukur waktu pengucapan agar sesuai dengan kebutuhan fungsionalitasnya.

2.2. Dasar Teori

Dasar teori yang dibutuhkan untuk penyusunan skripsi ini adalah Suara, Pengenalan Suara, Metode *Hidden Markov Model* dan PocketSphinx.

2.2.1. Pengenalan Suara

Pengenalan suara merupakan kemampuan sebuah alat/komputer dalam menerjemahkan suara menjadi bentuk teks ataupun komando. Teknologi ini mempunyai fasilitas untuk mengubah input suara kedalam rangkaian kosakata

dalam bentuk sinyal-sinyal suara lalu menghasilkan probabilitas model statistik untuk menemukan persamaan terhadap input suaranya [MSD-12].

Secara umum pengenalan suara dibagi 2 jenis diantaranya, pengenalan pembicara dan pengenalan ucapan. Pengenalan pembicara merupakan proses untuk mengenali siapa yang sedang berbicara atau mengucapkan informasi berdasarkan pola suara yang diinputkan. Teknologi ini sangat ampuh untuk mengidentifikasi suara seseorang dan bermanfaat untuk digunakan pada layanan seperti *control smarthome*, keamanan, layanan informasi dan akses terhadap keadaan tertentu. Jenis yang kedua yaitu, pengenalan ucapan yang didefinisikan sebagai proses pengubahan sinyal suara menjadi dalam bentuk teks. Pengenalan ucapan ini memiliki kemampuan untuk mencocokkan suara terhadap data yang ada seperti rekaman ataupun pembendaharaan kata [AMT-11]. Biasanya pengenalan ucapan digunakan untuk mengartikan ucapan seseorang ke dalam bentuk kata misalnya untuk penggunaan *voice control* pada *smarthome system* maupun sebagai sumber informasi untuk komunikasi kaum lansia dan disabilitas.

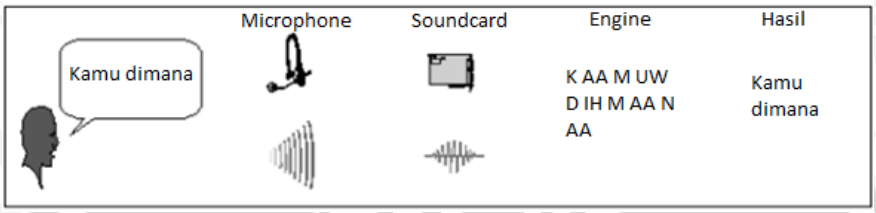
Pengenalan suara bekerja untuk menerjemahkan ucapan seseorang berdasarkan data suara yang sudah dimiliki oleh aplikasi dikarenakan teknologi ini dapat bekerja apabila sudah mengenali pola suara seseorang, makin banyak data suara yang yang dimiliki, makin bagus juga teknologi ini dalam mengenali suara seseorang.

Hasil dari identifikasi kata yang diucapkan dapat ditampilkan dalam bentuk kosakata maupun dapat dibaca oleh perangkat teknologi sebagai sebuah perintah untuk melakukan suatu pekerjaan, misalnya penekanan tombol pada kontrol *smarthome system* yang dilakukan secara otomatis dengan perintah suara (voice command).

2.2.1.1.Sistematika Pengenalan Suara

Sistem pengenalan ucapan adalah sistem yang berfungsi untuk mengubah bahasa lisan menjadi bahasa tulisan. Masukan sistem ini adalah ucapan manusia, selanjutnya sistem akan mengidentifikasikan kata atau kalimat yang diucapkan dan menghasilkan teks yang sesuai dengan apa yang diucapkan. Pada gambar 2.1 merupaka ilustrasi dasar pengucapan suara yang diolah oleh

komputer. Pertama kali suara dimasukkan menggunakan *microphone* lalu diterima oleh *soundcard* komputer kemudian diolah pada aplikasi/*library* pengenalan suara yang nantinya dikomputasikan sehingga dapat memunculkan kata yang diucapkan.



Gambar 2.1 Dasar Pengucapan Suara [FLC-14]

Penganalisis sintaks akan melakukan transformasi sinyal ucapan dari domain waktu ke domain frekuensi. Pada domain frekuensi, untuk kurun waktu yang singkat, setiap sinyal dapat terlihat memiliki ciri-ciri yang unik. Namun biasanya, pengucapan yang diucapkan oleh seseorang seringkali bervariasi yakni, terpengaruh oleh fonem-fonem, kondisi emosi, noise dan faktor-faktor lainnya. Sistem *speech recognition* akan melakukan pengenalan untuk setiap unit bunyi pembentuk ucapan (fonem), selanjutnya mencoba mencari kemungkinan kombinasi hasil ucapan yang paling dapat dikenali.

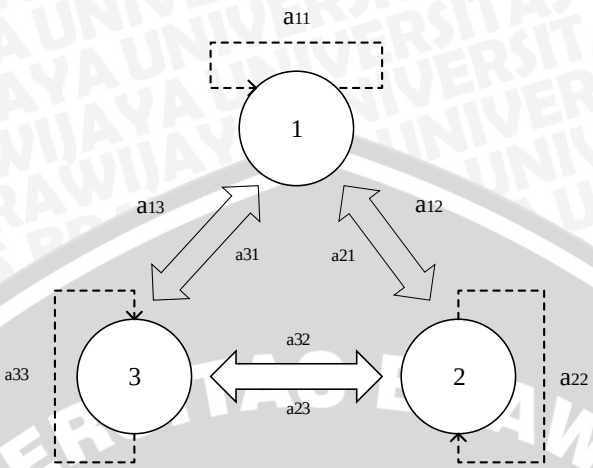
Sinyal ucapan pertama kali akan dilewatkan pada bagian penganalisis ucapan untuk mendapatkan besaran-besaran atau ciri-ciri yang mudah diolah pada tahap berikutnya. Untuk setiap ucapan yang berbeda akan dihasilkan pola ciri yang berbeda [MUB-10].

2.2.2. Hidden Markov Model

Model Markov ditemukan oleh Andrey Markov dan merupakan bagian dari proses stokastik. Model ini memiliki kehandalan untuk memprediksi keadaan yang akan datang artinya kondisi saat ini menangkap semua informasi yang mempengaruhi evolusi dari suatu sistem dimasa depan.

Model ini merupakan dari *finite automation* yaitu beberapa kumpulan state yang transisi antar *state*-nya dilakukan berdasarkan masukan observasi. Pada proses *Markov*, setiap busur antar state berisi probabilitas yang mengindikasikan

kemungkinan dari jalur tersebut. Jumlah probabilitas semua jalur yang keluar dari sebuah simpul memiliki nilai total satu [EBP-10].



Gambar 2.2 Rantai Markov [MVE-12].

Pada Gambar 2.2, misalnya dari simpul nomer 1 dan 2 memiliki kemungkinan a_{21} dan a_{12}. Maka jumlah probabilistic a_{12} dan a_{21} memiliki nilai satu. Hal ini juga berlaku pada simpul lainnya. Proses Markov memiliki manfaat untuk menghitung probabilitas suatu kejadian yang teramati dan secara umum dapat dirumuskan sebagai berikut:

$$P(q_1)[q_t = j|q_{t-1} = i, q_{t-2} = k\ldots] = P[q_t = j \mid q_{t-1} = i] \quad (2\text{-}1)$$

$\boldsymbol{q_1}$ adalah kondisi saat ini, dan $\boldsymbol{q_t}$ adalah kondisi pada waktu tertentu yang berhubungan dengan $\boldsymbol{q_1}$. Sedangkan $\boldsymbol{q_{t-1}}$ adalah kondisi sebelum $\boldsymbol{q_t}$. Kemudian kita dapat menyimpulkan bahwa keseluruhan sistem, transisi diantara keadaan tertentu tetap sama dengan hubungan probabilistiknya.

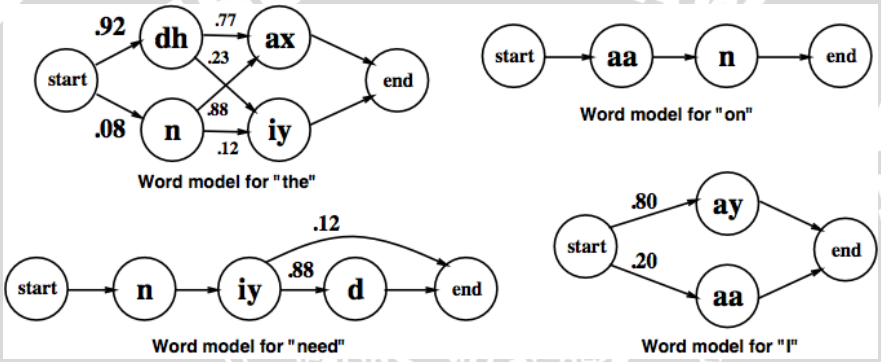
Berdasarkan persamaan pada paragraph sebelumnya, kita dapat membuat persamaan lainnya dimana persamaan tersebut memiliki transisi keadaan bebas terhadap waktu.

$$a_{ij} = P[q_t = j|q_{t-1} = i \quad (2\text{-}2)$$

Lahirnya metode *Hidden Markov Model* menjadi salah satu pemicu pembaruan teknik pengolahan speech (ucapan / suara) yang dapat menghasilkan

tingkat kesalahan seminimal mungkin dan meningkatkan ketangguhan pengklasifikasian pola pada berbagai kondisi yang kurang baik.

Penerapan HMM pada pengenalan suara berfungsi untuk memprediksi suara yang diucapkan berdasarkan *dictionary file*, *language model*, dan *acoustic model*nya. HMM akan memberikan probabilitas berdasarkan kosakata yang akan diucapkan. Probabilitas lalu dihitung berdasarkan pembobotannya sehingga mempunyai nilai terbaik yang memungkinkan kata tersebut dikeluarkan berdasarkan persamaan suara. Pencarian probabilitas terbaik dari masing-masing suku kata diselesaikan dengan Algoritma *Viterbi*.



Gambar 2.3 Pencarian Kata Terbaik dengan HMM [CMU-14]

Perbedaan model kata "the" dengan "need" tergantung kesamaan suku kata yang dimiliki kata tersebut di dalam *dictionary file*. Apabila kata pada *dictionary file* memiliki suku kata yang hampir sama dengan kata yang lain maka percabangan HMM akan semakin banyak sehingga memiliki banyak kemungkinan dari pembentukan kosakata berdasarkan suku katanya. Jika masalah ini terjadi HMM hanya dapat memprediksi kemungkinan berdasarkan pola suara yang paling sering diucapkan oleh seseorang.

2.2.3. PocketSphinx

Pocketsphinx ialah sistem pengenalan suara tercepat yang dikembangkan oleh Carnegie Mellon University. Pocketsphinx menggunakan bahasa pemrograman C murni dan sangat optimal karena bersifat *real-time* sehingga mesin dapat mengenali suara dengan akurat. Dengan mengandalkan respon yang sangat cepat dan konsumsi sumber daya yang minim sehingga memungkinkan

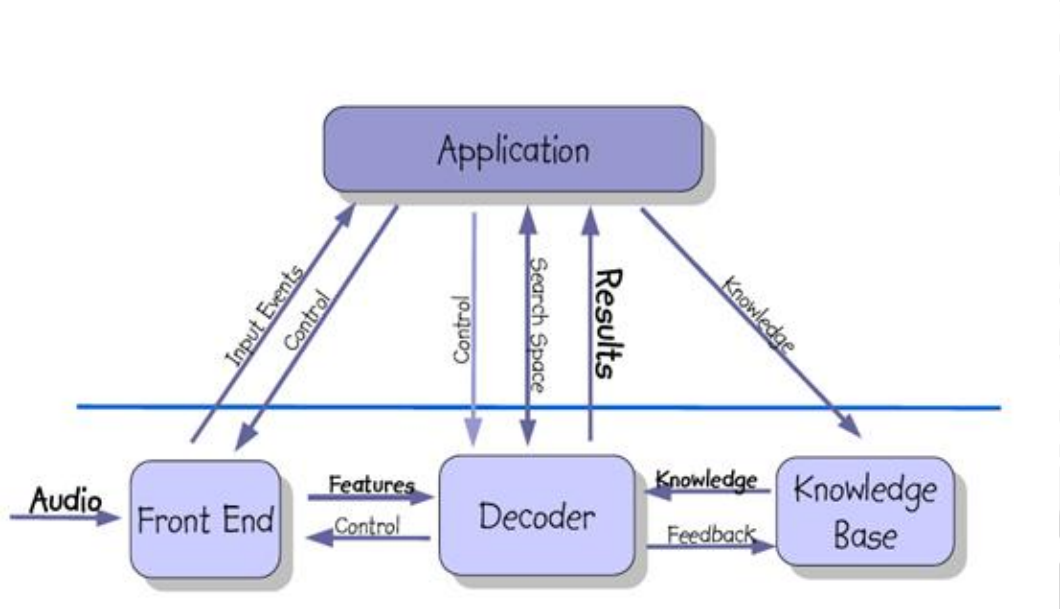
pengembangan pengenalan suara cocok digunakan untuk aplikasi desktop, komando dan kontrol serta pendiktean. Selain itu, *engine* ini dapat berguna bagi perangkat tertanam dengan *fixed-point ariphmetics* yang berhasil digunakan pada iPhone, perangkat Nokia Maemon dan Windows Mobile [CMU-14].

Library ini menyediakan fasilitas untuk para penggunaanya untuk memodifikasi Bahasa yang akan digunakan untuk mengenali suara seseorang. Aplikasi PocketSphinx merupakan sistem pengenalan suara yang biasa digunakan untuk aplikasi desktop. Pada saat ini, PocketSphinx biasa digunakan untuk kebutuhan *embedded system* dan robot karena library didukung memiliki kelebihan komputasi yang cepat dan ringan dalam mengenali suara seseorang.

2.2.3.1.Sejarah Perkembangan PocketSphinx

Berdasarkan sejarah perkembangan teknologi pengenalan suara pada tahun 70an Carnegie Mellon University dan perusahaan IBM ShoeBox bekerja sama untuk membuat aplikasi pengenalan suara. Selanjutnya pada tahun 2000, kelompok Sphinx di Carnegie Mellon berkomitmen untuk membuat *open source* yang dapat mengenali suara oleh komputer, diantaranya Sphinx 2 dan kemudian Sphinx 3 dikeluarkan pada tahun 2001. Decoder *speech recognition* mengeluarkan dengan model akustik dan contoh aplikasi. *Resource* yang tersedia termasuk dalam perangkat lunak tambahan untuk model *acoustic model* yang menggunakan *SphinxTrain* yang merupakan *library* untuk pelatihan suara, *language model* dan *dictionary file*. Berbagai macam decoder *speech recognition* tersedia dengan berbagai bahasa pemrograman diantaranya PocketSphinx menggunakan bahasa C dan Sphinx menggunakan bahasa pemrograman *java* [CMU-14].

2.2.3.2.Arsitektur Sphinx



Gambar 2.4 Arsitektur Sphinx [CMU-14]

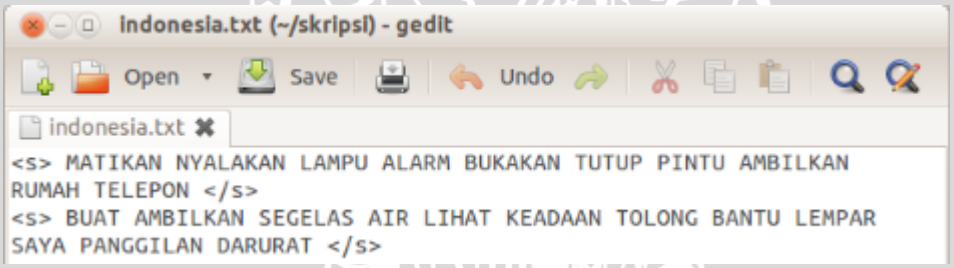
Gambar 2.4 menunjukan arsitektur umum yang digunakan oleh *Sphinx*. Setiap elemen pada gambar ini dapat diganti sesuai dengan kebutuhan peneliti. Dengan demikian, kita dapat mengubah beberapa fitur tanpa mengubah sisi fungsionalitasnya secara umum.

Arsitektur umum aplikasi ini terdiri komponen umum untuk menunjang hasil yang presisi pada saat pengujian aplikasi. Kompenen tersebut terdiri dari *front end*, *decoder*, basis pengetahuan, dan aplikasi itu sendiri. *Front end* bertanggung jawab untuk mengumpulkan, memberikan keterangan dan pengolahan input data. Selain itu, *front end* juga menyediakan kemudahan untuk pengaksesan API audio yang merupakan fitur untuk merekam masukan pengguna ucapan dan memutar hasil rekaman tersebut. Pada basis pengetahuan memberikan sebuah informasi decoder untuk melakukan tugasnya. Informasi tersebut mencakup model akustik dan model bahasa. Basis pengetahuan pula yang memberikan umpan balik dari decoder sehingga memungkinkan basis pengetahuan dapat dimodifikasi secara dinamis. Sedangkan pada decoder sendiri berfungsi sebagai komponen utama pada aplikasi pengenalan suara. Decoder berfungsi untuk menentukan urutan yang paling mungkin dari kata-kata yang dapat diwakili oleh serangkaian fitur dan dilakukan secara dinamis selama proses *testing/decoding* [CMU-14].

2.2.3.3.*Language Model*

Language Model merupakan pembangkit *grammar* kosakata pada aplikasi pengenalan suara. Kompleksitas *grammar* tergantung pada sistem yang akan dikembangkan. Dalam penelitian kali ini, *Language Model* dibangun berdasarkan statistic yang digunakan oleh *library* PocketSphinx. Toolkit *Language Model* dibuat berdasarkan pemodelan *uni-gram*, *bi-gram*, dan *tri-gram* dari Bahasa yang akan dikenali. Penciptaan *Language Model* terdiri dari komputasi kata jumlah *uni-gram* yang kemudian diubah menjadi kosakata dengan frekuensinya sehingga nantinya dapat menghasilkan *bi-gram* dan *tri-gram* dari teks pelatihan dan akhirnya mengubah *n-gram* ke dalam format biner *Language Model* dan format ARPAbet. Pada penelitian kali ini, *Language Model* dibuat dengan cara menuliskan data artikel yang ingin dikenali oleh aplikasi *voice command* menggunakan notepad lalu menguploadnya ke server CMU Sphinx. Pada gambar 2.3 merupakan pembuatan file *Language Model* [MAM-10]

.



Gambar 2.5 Pembuatan Data Artikel

2.2.3.4.*Acoustic Model*

Acoustic Model merupakan fasilitas yang diberikan oleh PocketSphinx untuk mengenali ciri suara pengguna. *Acoustic Model* menggunakan model *Hidden Markov Model* dan *Gaussian Mixture* sebagai metode dalam menentukan sinyal suara yang disesuaikan dengan fonem seseorang. *Hidden Markov Model* pada *Acoustic Model* merepresentasikan beberapa parameter berupa *hidden state* yang berupa input suara, *observed state* berupa fitur vektor *input* suara, dan probabilitas transisi yang berupa nilai probabilitas terbaik untuk pengucapan suara sehingga dapat membangkitkan ciri suara yang digunakan pada aplikasi.

Pelatihan *Acoustic Model* dilakukan dengan cara merekam suara yang disesuaikan dengan *transcription file* dari setiap kalimat yang diucapkan dalam rekaman suara. Inputan *Acoustic Model* berupa rekaman suara dengan format *.wav*. Setiap file berisikan satu kalimat dari satu pembicara. Selanjutnya rekaman suara tersebut disesuaikan dengan *transcription file.* Pedoman pada *transcription file* merujuk *Dictionary file. Filler file* yang berisikan suara *non-speech* yang bertujuan memberikan jeda pada sebelum kalimat mulai dan berakhir diucapkan [ACM-14].

2.2.3.5.*Dictionary File*

Dictionary file berisikan kosakata yang digunakan dalam teks artikel dan cara-cara pengucapan tiap kosakata yang terdiri dari fonem-fonem yang dapat disesuaikan dengan model Bahasa yang akan digunakan. Pada keadaan awal, teks artikel diupload ke server CMU Sphinx menggunakan web browser dengan alamat http://www.speech.cs.cmu.edu/tools/lmtool-new.html. Setelah file tersebut diupload, maka akan dihasilkan *dictionary file* yang berisikan kosakata yang dapat diucapkan oleh pengucapan bahasa Inggris. Oleh karena itu, fonem-fonem harus disesuaikan dengan gaya pengucapan orang Indonesia [NKA-10].

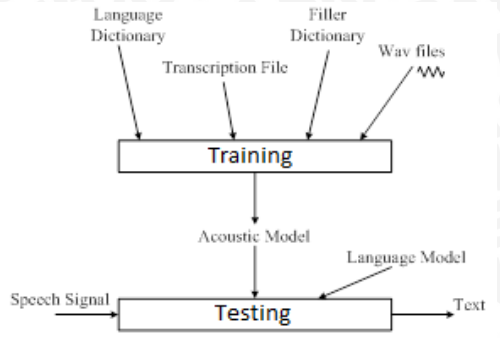
Dictionary file menjadi pedoman utama untuk pengenalan suara yang akan dirancang karena kosakata yang diucapkan terbatas sesuai dengan kosakata yang ada di *Dictionary file*. Penulisan kosakata pada *Dictionary file* menggunakan peraturan Arpabet. Arpabet adalah transkripsi fonetik yang dikembangkan oleh *Advanced Research Projects Agency* (ARPA) sebagai bagian dari Speech Understanding Project (1971-1976).Arpabet mewakili setiap fonem dari dalam bahasa Inggris Amerika umum (General American English) dengan urutan yang berbeda dari karakter ASCII. Peraturan Arpabet dapat dilihat pada **Lampiran 1** [MAM-10].

2.2.3.6.*Training*

Training merupakan upaya pelatihan suara untuk membangkitkan kosakata yang akan diucapkan oleh seseorang. *Training* berfungsi sebagai pengenalan ciri suara manusia sehingga nantinya suara dapat dikenali dengan baik. *Training* akan mempelajari model unit suara menggunakan kumpulan sampel sinyal suara pada database *Training* berdasarkan *transcription file*. Selain *transcription file*, dibutuhkan juga kamus pengucapan (dictionary file) yang memetakan setiap kata ke urutan unit suara. Hasil *Training* menghasilkan akurasi rekaman suara yang telah dipetakan pada database *Training*, hasil akurasi yang terbaik dapat didapatkan apabila nilai parameter-parameter pada saat *Training* digunakan secara maksimal. Setelah proses *Training* berakhir dengan menghasilkan hasil akurasi yang baik lalu peneliti dapat mengambil file *acoustic model* yang akan dapat digunakan untuk membantu *testing* pengenalan suara [MVE-12]. Proses training suara menggunakan *library* dari CMU Sphinx yaitu SphinxTrain. *Training* membantu untuk membangkitkan probabilitas suatu kosakata, apabila kosakata sering dilatih maka akurasi kosakata yang diucapkan pada saat *testing* akan semakin baik.

2.2.3.7.*Testing*

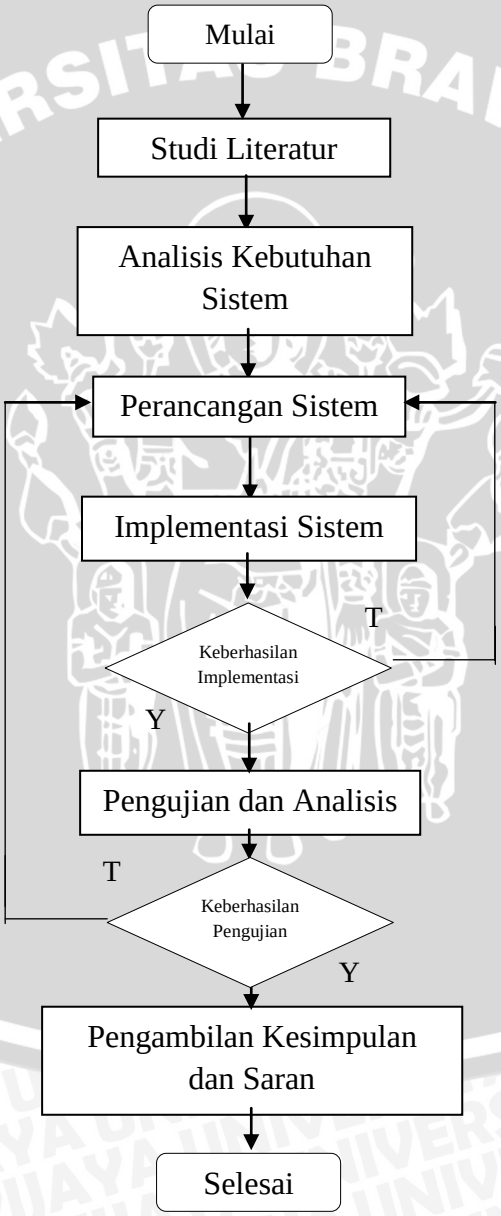
Testing merupakan percobaan pengenalan suara dengan memberikan *executable* tunggal yang dapat melakukan tugas pengenalan suara saat diberi inputan suara. File-file yang dibutuhkan pada saat melakukan testing ialah *acoustic model*, *dictionary file*, dan *language model*. Testing dapat dilakukan dengan baik apabila proses pada saat *training* dilakukan dengan baik untuk mengenali pola suara orang Indonesia. Pada percobaan *testing* menggunakan library PocketSphinx [MVE-12].



Gambar 2.6 Hubungan antar Proses *Training* dan *Testing* [CSA-10]

BAB III
METODE PENELITIAN DAN PERANCANGAN

Bab ini menjelaskan langkah-langkah yang dilakukan dalam penyusunan skripsi, yaitu studi literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis serta pengambilan kesimpulan dan saran sebagai referensi untuk pengembangan perangkat lunak selanjutnya. Berikut ini merupakan diagram alir tahap pengerjaan penelitian ini :



Gambar 3.1 Diagram Alir Penelitian

3.1. Studi Literatur

Dalam perancangan dan implementasi penelitian ini, perlu diadakan studi literatur. Studi literatur menjelaskan dasar teori yang digunakan sebagai penunjang dan pendukung penulisan skripsi. Teori penunjang dan pendukung tersebut didapat dari buku, jurnal, paper dan internet. Literatur yang digunakan meliputi:

a. Pengenalan Suara
b. Hidden Markov Model
c. PocketSphinx

3.2. Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem pada aplikasi yang akan dibangun dan diuji. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan *(requirements)* sistem. Analisis juga dilakukan untuk mengetahui kondisi lapangan yang ada sehingga dapat diketahui implementasi perangkat lunak dan perangkat keras yang akan digunakan. Analisis kebutuhan ini didapat berdasarkan berbagai sumber di internet, buku-buku, dan jurnal-jurnal tentang definisi *speech recognition* untuk menunjang pembuatan aplikasi pengenalan *voice command* berbahasa Indonesia.

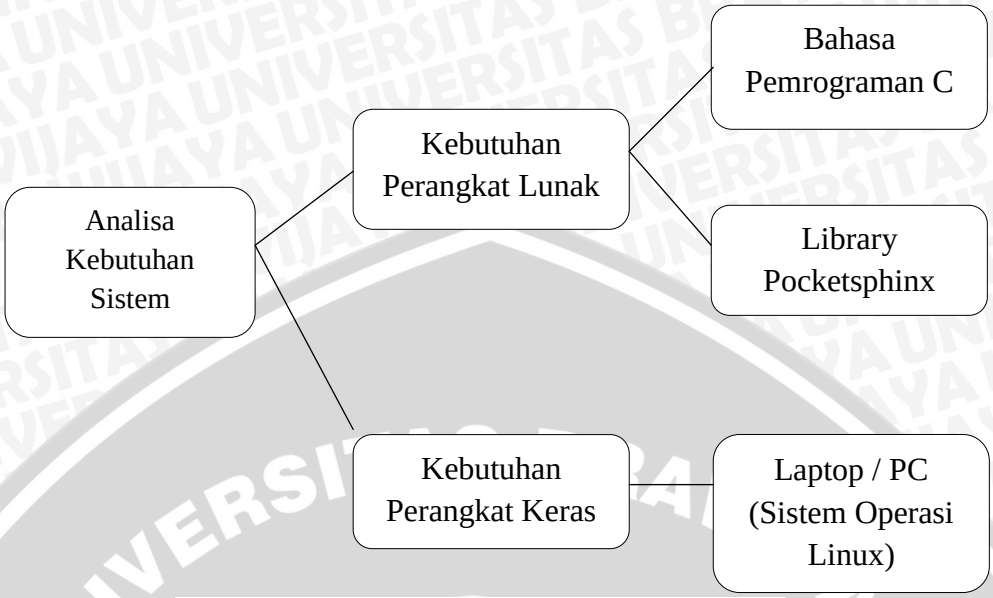
Proses analisis kebutuhan ini diawali dengan penjabaran gambaran umum pembangunan aplikasi "Pengenalan *Voice Command* Berbahasa Indonesia". Penjabaran gambaran umum ini dilakukan untuk mendeskripsikan kebutuhan-kebutuhan dan fungsionalitas sistem pada perancangan skripsi ini.

Pada penelitian ini, untuk merancang sistem *voice command speech to text*, dibutuhkan sebuah pengujian pada sebuah komputer terlebih dahulu untuk melakukan pelatihan terhadap aplikasi *voice command speech to text*. Selain tempat untuk pengujian sistem, dibutuhkan juga lingkungan sistem yang sesuai bagi aplikasi sehingga nantinya dapat dengan mudah diterapkan pada *smarthome voice control*. Aplikasi *voice command speech to text* ini membutuhkan suatu lingkungan yang digunakan sebagai tempat berjalannya aplikasi. Secara keseluruhan aplikasi ini merupakan rintisan dalam perancangan bagi *smarthome voice control* tetapi pada kesempatan kali ini penulis hanya akan mencoba aplikasi

ini pada desktop komputer yang berbasis sistem operasi Linux. Hal ini berdasarkan kemudahan sistem operasi tersebut terhadap hubungan dengan mesin (*embedded system*). Peneliti memilih sistem operasi Linux sebagai lingkungan sistem yang memadai untuk pengujian aplikasi ini. Diharapkan setelah perancangan pada desktop selesai, aplikasi dapat diterapkan diberbagai macam lingkungan *device/hardware* yang membutuhkan fungsionalitas dari aplikasi ini.

Analisis kebutuhan lainnya yang dibutuhkan pada perancangan skripsi ini ialah kebutuhan perangkat lunak. Kebutuhan perangkat lunak merupakan hal yang mutlak untuk pembuatan sebuah aplikasi. Perangkat lunak ini menggunakan bahasa pemrograman C. Bahasa pemrograman ini cocok untuk perancangan yang terkait dengan aplikasi yang berhubungan dengan mesin. Bahasa pemrograman C ini dirancang dengan bantuan *library* PocketSphinx. Penggunaaan library ini sangat memudahkan peneliti untuk merancang aplikasi ini untuk pengenalan suara dengan berbagai bahasa secara *flexible*. Peneliti menginginkan dengan aplikasi dibangun ini dapat merancang *voice command speech to text* dalam Bahasa Indonesia.

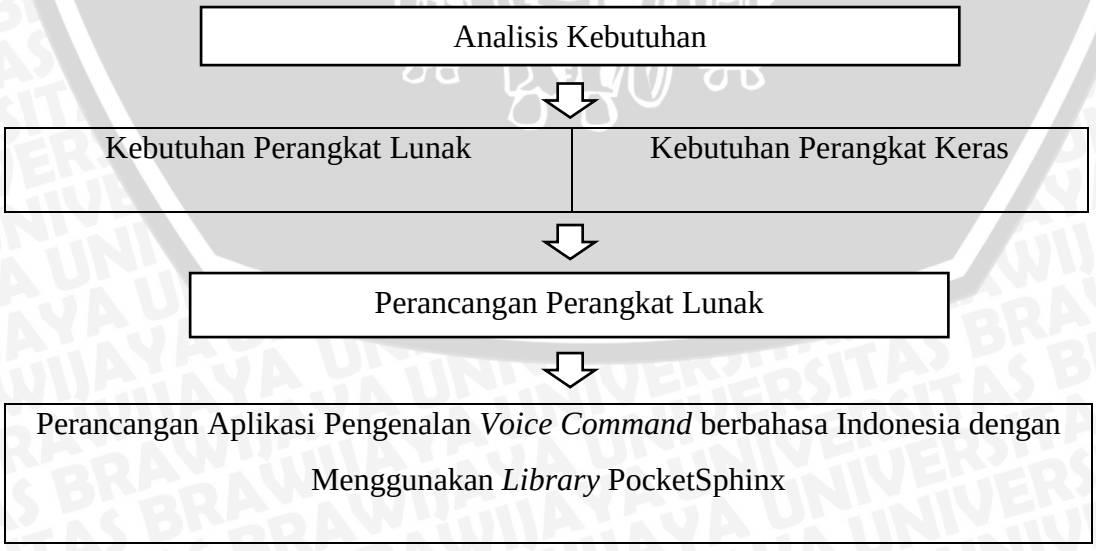
Library PocketSphinx merupakan *library* untuk pengenalan suara bagi Bahasa inggris, untuk mengubahnya menjadi pengenalan suara bagi bahasa Indonesia peneliti harus mengubah kamus pengucapan yang disesuaikan dengan fonem orang Indonesia dan melakukan *training* suara dengan menggunakan koresponden yang berbahasa Indonesia supaya *library* dapat mengenali suara bahasa Indonesia. Gambaran umum mengenai kebutuhan sistem pada penelitian kali ini dapat dilihat pada gambar 3.2.

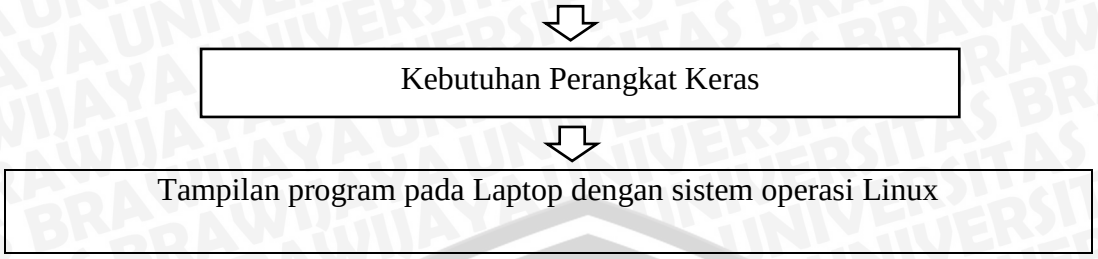


Gambar 3.2 Diagram Kebutuhan Sistem

3.3. Perancangan Sistem

Setelah semua kebutuhan perangkat lunak diperoleh dari tahap analisis kebutuhan tahap selanjutnya adalah perancangan perangkat lunak. Perancangan perangkat lunak berdasarkan pada diagram alir yang telah dibuat. Perancangan dimulai dari perancangan alur atau aktifitas yang dilakukan user secara prosedural. Berikut ini ialah tahapan-tahapan perancangan sistem aplikasi pengenalan *voice command* berbahasa Indonesia:





Gambar 3.3 Perancangan Sistem Secara Umum

3.4. Implementasi

Implementasi perangkat lunak mengacu kepada perancangan perangkat lunak. Implementasi perangkat lunak diawali dengan penjabaran spesifikasi lingkungan perancangan perangkat lunak. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman C dengan menggunakan *library* PocketSphinx. Pada tahap *training* hingga *testing* dilakukan pada sistem operasi Linux. Proses ini mungkin tidak berjalan satu kali, Jika dirasa program belum memenuhi kebutuhan, aplikasi akan didesain ulang dalam perangkat lunak.

Implementasi di penelitian ini dibagi 2 tahap yaitu, proses *training* yang bertujuan untuk melatih suara orang Indonesia sehingga dapat mengenali pola suaranya dan proses *testing* yang bertujuan untuk mengetahui keakuratan sistem dalam mengenali bahasa Indonesia. Proses *training* memiliki persyaratan tingkat keakuratan sistem sehingga dapat mengenali pola suara orang Indonesia dengan baik. Proses *training* dilakukan dengan cara merekam suara orang Indonesia yang disesuaikan dengan *transcription file* yang berupa kumpulan kalimat dari Bahasa Indonesia. *Training* dilakukan dengan menggunakan *library* SphinxTrain. *Training* dilakukan hingga memunculkan hasil terbaik dan dilakukan analisis sehingga dapat diukur dan dapat dituangkan ke dalam laporan skripsi ini. Setelah hasil pengujian *training* sesuai dengan persyaratan tingkat keakuratan sistem, dapat dilakukan analisis sehingga nantinya dapat digunakan pada saat proses *testing*. Pada saat proses *testing*, hasil proses *training* yang berupa *acoustic model* dijadikan file pendukung selain *dictionary file* dan *language model*. Implementasi terhadap *testing* dan *training* merupakan perintisan untuk melakukan pengujian sehingga diketahui hasil akurasi dari kedua proses tersebut dalam mengenali suara bahasa Indonesia.

Hardware yang digunakan selama pembuatan aplikasi pembelajaran bahasa Indonesia hanya digunakan 1 buah laptop. Sistem operasi komputer yang digunakan adalah Linux Ubuntu 12.04.

3.5. Pengujian dan Analisis

Pengujian aplikasi akan menganalisis pengujian validitas/keakuratan dari hasil pengenalan suara bahasa Indonesia dan hasil *training* suara berdasarkan pengubahan parameter maupun dari rekaman 7 koresponden suara. Pengujian perangkat lunak pada skripsi ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang telah dianalisa sebelumnya.

Pengujian validitas dilakukan untuk mengetahui hasil keakuratan pengenalan suara dengan menggunakan Bahasa Indonesia. Selanjutnya, pengujian pengenalan *voice command* berbahasa Indonesia dilakukan pengujian menggunakan *testing* dengan membandingkan keberhasilan sistem dengan menggunakan *parameter-parameter* penentu keberhasilan dalam membaca kata-kata yang diucapkan oleh *user*. Selanjutnya, pengujian terhadap waktu dilakukan sebagai informasi sehingga nantinya dapat diterapkan pada *smarthome system* secara *real-time*. Keberhasilan tersebut dicatat dan dihitung rata-rata total akurasinya yang dituangkan ke dalam laporan skripsi. Analisis juga dilakukan untuk mengetahui hasil dari pengujian perangkat lunak sehingga dapat didapatkan kesimpulan dan pengembangan perangkat lunak yang telah dilakukan.

Pengujian *testing* aplikasi dilakukan oleh penulis (1 orang) dan pengguna (4 orang) serta tersedia 25 kosakata dalam bahasa Indonesia. Hasil pengujian yang didapat adalah hasil keakuratan yang baik, dan dapat digunakan secara *offline* (tanpa menggunakan akses internet).

3.6. Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak telah diselesaikan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Kesimpulan diambil untuk mengetahui kesesuaian program

terhadap perancangan yang telah dibuat. Selanjutnya, peneliti melakukan perbaikan pada kesalahan-kesalahan yang terjadi pada pembuatan aplikasi ini. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta sebagai acuan untuk pengembangan perangkat lunak selanjutnya.

3.7. Perancangan

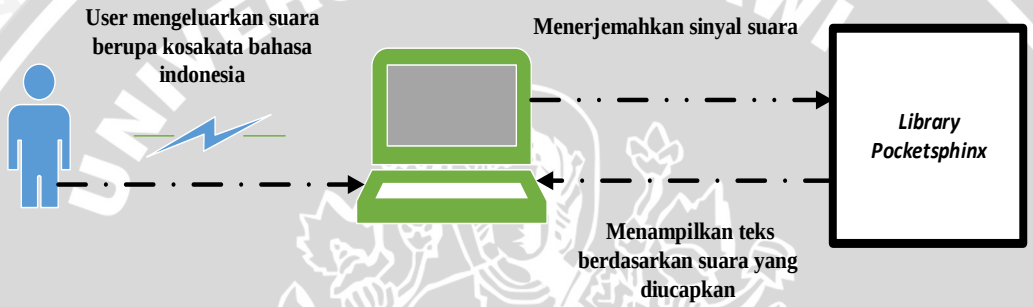
Pada proses perancangan aplikasi *voice command speech to text* terdapat 4 (empat) tahapan yaitu, tahap pertama adalah gambaran umum aplikasi, tahap kedua adalah perancangan aktifitas, tahap ketiga adalah perancangan *training* suara serta yang terakhir yaitu tahap keempat adalah proses perancangan *testing* kosakata dalam bahasa Indonesia. Tahap gambaran umum sistem menjelaskan mengenai gambar proses cara kerja sistem menggunakan *library* PocketSphinx. Tahap aktifitas merepresentasikan aktifitas yang terjadi pada saat melakukan perancangan aplikasi untuk pengenalan suara bahasa Indonesia secara keseluruhan. Pada proses perancangan untuk *training* suara ialah sarana bagi program untuk mengidentifikasi kata-kata yang diucapkan oleh orang Indonesia dengan menggunakan metode *Hidden Markov Model*. Pada tahap perancangan *testing* program ialah proses dari awal pada saat user menginputkan suara lalu diproses oleh aplikasi hingga menghasilkan output yang berupa data *text*. Setelah perancangan aplikasi ini selesai maka dilakukan analisa pada tahap pengujian / *training* kata yang meliputi pengumpulan data, pelatihan parameter (kata) dan penyimpanan hasil *training*.

3.7.1. Gambaran Umum Aplikasi

Aplikasi *voice command speech to text* untuk bahasa Indonesia merupakan suatu media yang menggunakan metode pendengaran untuk peningkatan kemampuan bahasa. Tujuan dari pembuatan aplikasi ini untuk membantu para kaum disabilitas yang membutuhkan aplikasi ini untuk media komunikasi diantara mereka. Selain digunakan untuk kaum disabilitas, aplikasi ini juga digunakan untuk kebutuhan smarthome berbasis *voice control*.

Selama penggunaan aplikasi, tidak dibutuhkan adanya koneksi internet agar dapat menjangkau di berbagai tempat tanpa terkendala koneksi internet.

Aplikasi ini diharapkan dapat diterapkan pada *smarthome voice control* dan dapat membantu para kaum disabilitas dan lansia yang membutuhkan fungsi aplikasi ini. Hanya, pada kesempatan kali ini peneliti hanya mengimplementasikan teknologi pada laptop yang menggunakan sistem operasi linux ubuntu 12.04. Diharapkan dengan pengimplementasian pada laptop dapat memaksimalkan pelatihan suara dengan Bahasa Indonesia sehingga nantinya dapat diimplementasikan kepada perangkat keras lainnya yang menunjang teknologi berbasis *voice control*. *Library* PocketSphinx mengolah suara yang diucapkan oleh user seperti digambarkan oleh gambar 3.4.

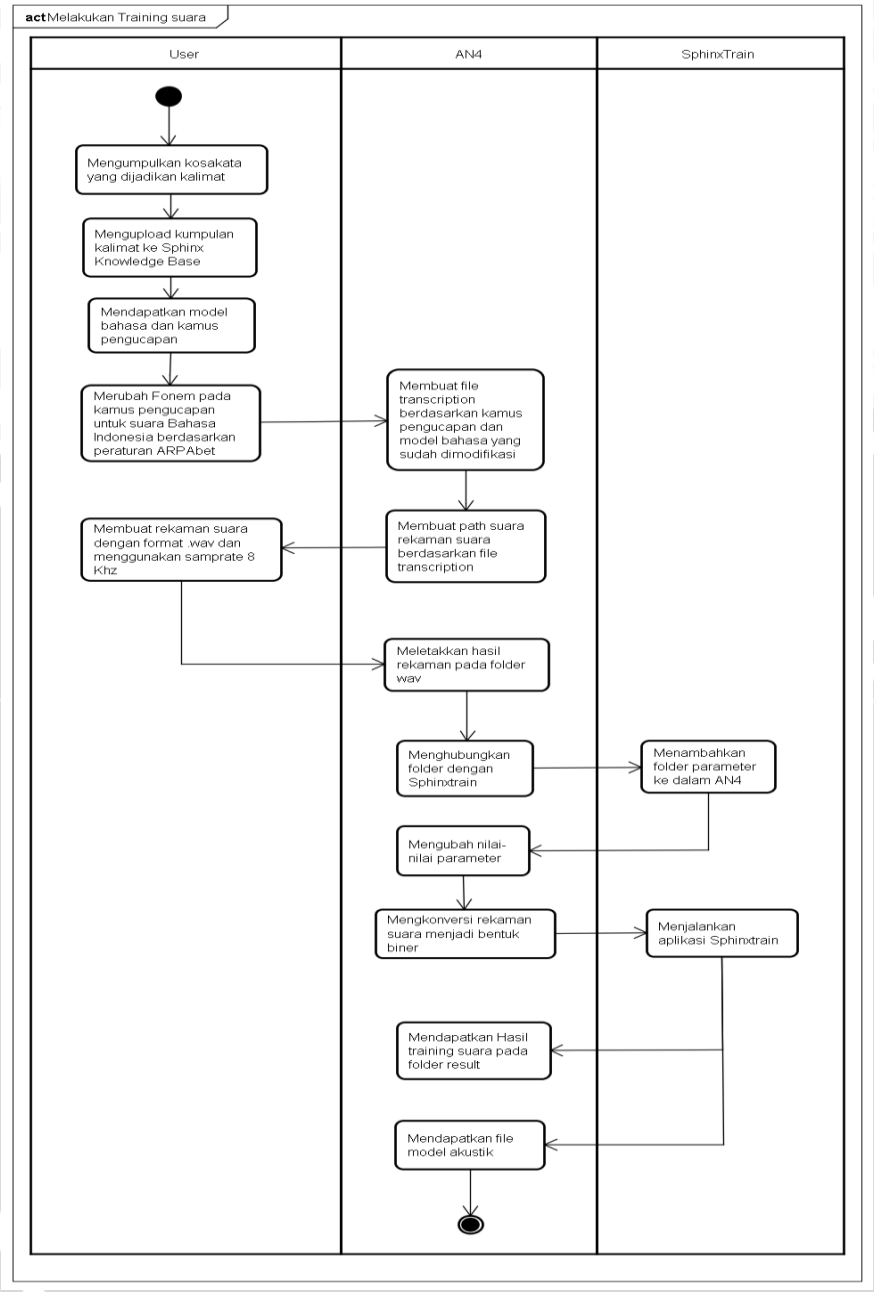


Gambar 3.4 Gambaran Umum Aplikasi

3.7.2. Perancangan Aktifitas

Perancangan aktifitas merupakan representasi dari aktifitas yang dilakukan oleh peneliti pada saat melakukan aktifitas perancangan aplikasi *voice command speech to text*. Aktifitas tersebut diwakili dengan diagram *activity*. Diagram *activity* adalah diagram yang merepresentasikan aktifitas yang terjadi pada saat melakukan perancangan aplikasi.

1. *Activity* Diagram *Training* Suara

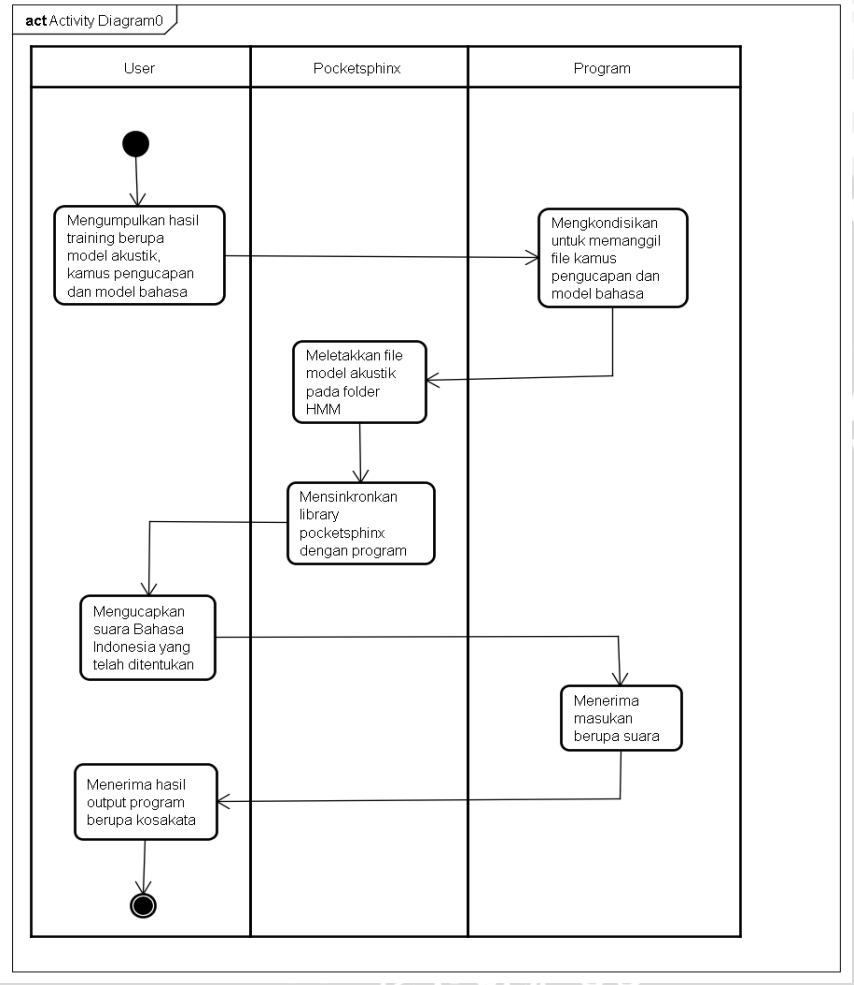


Gambar 3.5 Activity Diagram untuk *Training* Suara

Gambar 3.5 menjabarkan skenario proses yang terjadi pada saat implementasi *training* suara menggunakan bahasa Indonesia. Pada saat melatih suara harus menggunakan library *SphinxTrain* dan AN4 yang berfungsi sebagai *database* rekaman suara dan *file transcription*nya. Rekaman suara menggunakan format berbasis *.wav* yang disesuaikan dengan *file transcription*nya. Selain

menyiapkan *file transcription* yang disesuaikan dengan rekaman suara, peneliti juga harus menyiapkan kamus pengucapan sebagai pedoman fonem suara bahasa Indonesia.

2. *Activity* Diagram *Testing* Suara



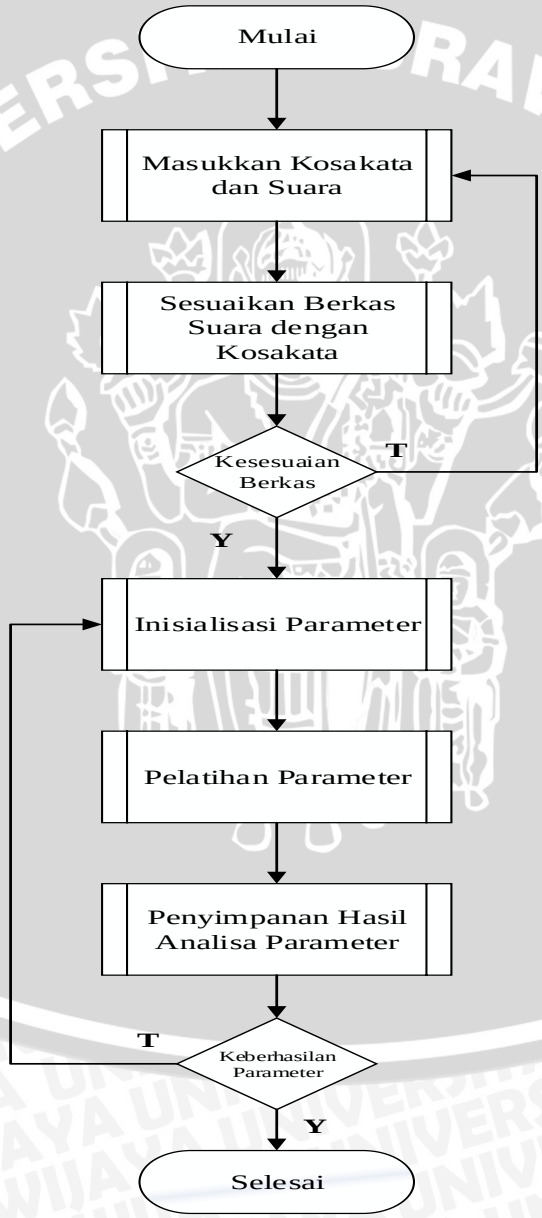
Gambar 3.6 Activity Diagram untuk *Testing* Suara

Gambar 3.6 menjabarkan skenario untuk melakukan *testing* suara bahasa Indonesia dengan menggunakan *library* PocketSphinx. Pada tahap *testing* suara, terlebih dahulu melakukan *training* suara untuk mendapatkan model akustik (acoustic model) bahasa Indonesia. Selanjutnya, meletakkan kamus pengucapan dan model bahasa yang disesuaikan dengan pemanggilan *file-file* tersebut pada program. Pada tahap akhir dari aktifitas ini, user akan mengucapkan suara bahasa

Indonesia secara langsung, lalu diterjemahkan oleh *library* sehingga menghasilkan output berupa kosakata.

3.7.3. Perancangan *Training* Suara

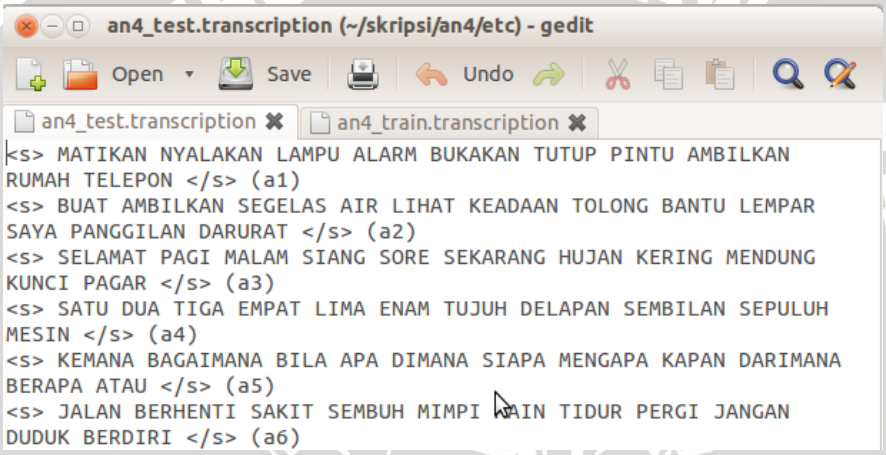
Training suara dilakukan untuk mengenali ciri suara dari beberapa warna suara (manusia). Pelatihan suara ini juga berguna untuk melakukan *testing* program supaya suara yang dikeluarkan oleh user dapat tepat menghasilkan kosakata yang user ucapkan. Pada gambar 3.7 merupakan ilustrasi perancangan *training* suara yang akan dilakukan.



Gambar 3.7 Diagram Proses *Training* Suara

a. Memasukkan Kosakata dan Suara

Peneliti mengumpulkan kosakata yang akan diucapkan oleh seseorang. Kosakata yang digunakan akan disesuaikan untuk perancangan *smarthome* dan teknologi yang nantinya dapat membantu kaum lansia dan disabilitas. Pada saat kosakata telah dimasukkan ke dalam *database training*, lalu direkam oleh suara yang telah disesuaikan. Untuk satu kali perekaman dilakukan oleh satu orang berdasarkan satu baris kalimat yang telah ditentukan. Gambar 3.8 merupakan kosakata yang akan diuji pada aplikasi *voice command speech to text*.



Gambar 3.8 Kamus kata yang akan dilatih

b. Sesuaikan berkas Suara dan Kosakata

Pada saat melakukan *training* sesuaikan berkas yang akan diminta pada program untuk melakukan *training*. Ketika berkas yang diminta tidak sesuai dengan proses *training* maka tidak bisa dijalankan. File yang dibutuhkan ialah rekaman suara yang berformat *.wav* dengan *samprate* 8000 Hz 16 bit, *dictionary file*, *language model*, *transcription file* yang disesuaikan dengan rekaman dan *transcription file path*nya, serta *fillers file*.

c. Inisialisasi Parameter

Pada tahap ini akan dilakukan inisialisasi parameter berdasarkan kata-kata yang akan diimplementasikan nanti. Parameter yang akan digunakan pada program nantinya ialah *lowerf, upperf, nfft dan nfilt*. Pemilihan parameter

ini digunakan sebagai dasar perhitungan untuk hasil latihan kata-kata yang sudah diimplementasikan kepada program.

d. Pelatihan Parameter

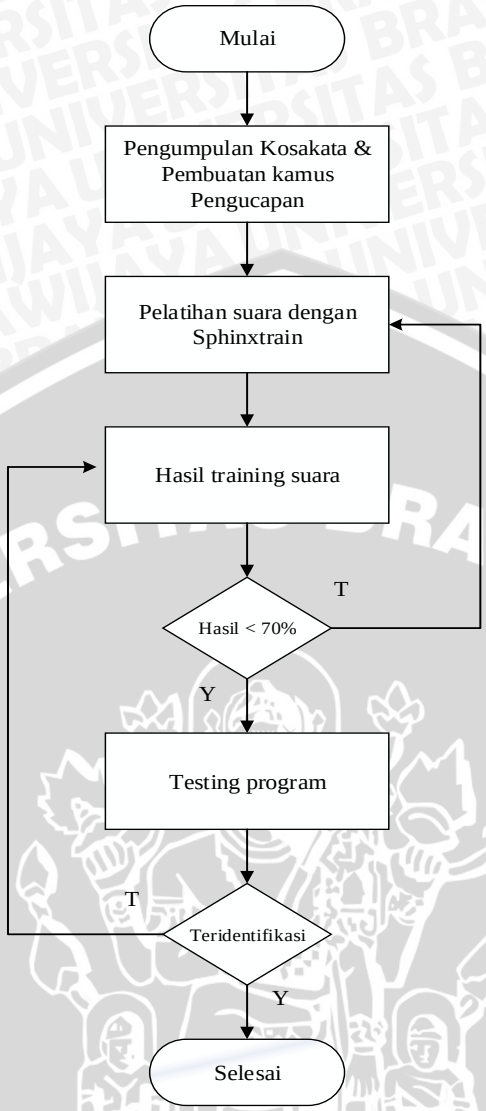
Dalam proses pelatihan ini parameter yang sudah didapatkan dari hasil inisialisasi parameter akan diestimasi sampai didapatkan *error* terkecil. Pelatihan parameter dilakukan dengan cara mengubah nilai *nfft/nfilt* dan *upperf/lowerf*. Nilai dari parameter-parameter tersebut diubah sehingga menghasilkan nilai akurasi terkecil pada *Word Error Rate*. Proses training suara dilakukan pada sistem operasi Ubuntu 12.04 LTS dengan *library* PocketSphinx.

e. Penyimpanan Hasil Analisa Parameter

Penyimpanan parameter diperlukan karena parameter ini akan berfungsi untuk proses pengenalan kata. Setelah proses training selesai dijalankan akan dihasilkan 4 file dengan modifikasi yang akan ditentukan sehingga menghasilkan hasil *error* terkecil yaitu lowerf, upperf, nfft dan nfilt. Beberapa file tersebut beserta daftar kata dan *language model* nantinya akan di push kedalam sistem yang digunakan untuk proses pengenalan suara dan dirubah menjadi *text*.

3.7.4. Perancangan *Testing* Program

Perancangan *testing* program berfungsi untuk mencoba program pengenalan suara ini sekaligus menunjukkan bahwa inputan program yang akan dimasukkan sama persis dengan output yang akan dikehendaki. Inputan suara disesuaikan dengan kamus pengucapan berdasarkan fonem bahasa Indonesia. Proses inputan suara ini dipengaruhi oleh hasil model akustik yang didapat dari hasil *training* suara. Dari hasil training suara tersebut menghasilkan model parameter yang digunakan sebagai optimasi terhadap pengenalan suara. Beberapa tahapan yang akan dilalui setelah suara yang diucapkan hingga menghasilkan output berupa teks terdapat pada gambar 3.9 yang merupakan proses tahapannya.



Gambar 3.9 Perancangan *Testing* Program

Pada gambar 3.9 menunjukkan bahwa inputan program akan dimasukkan berupa ucapan manusia dalam Bahasa Indonesia menggunakan *microphone* pada laptop/PC, lalu kosakata tersebut dilatih dengan beberapa model suara dengan menggunakan *SphinxTrain*. Pada tahap akhir setelah ucapan tersebut diolah maka diharapkan dapat mengeluarkan *output* pada layar monitor laptop berupa teks kosakata dalam Bahasa Indonesia.

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan perancangan yang sudah dipaparkan pada bab sebelumnya. Pembahasan pada bab ini terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi *training* suara, implementasi *testing* program dan implementasi program terhadap metode *Hidden Markov Model*.

4.1. Spesifikasi Sistem

Hasil dari analisis kebutuhan dan perancangan lunak yang telah dijelaskan pada tahap analisis kebutuhan dan tahap perancangan sistem menjadi dasar untuk dilakukan implementasi menjadi sebuah aplikasi pengenalan *voice command* Berbahasa Indonesia yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

4.1.1. Spesifikasi Perangkat Lunak

Dalam proses pengembangan aplikasi pengenalan *voice command* Berbahasa Indonesia ini menggunakan perangkat lunak dengan spesifikasi yang akan dijelaskan pada Tabel 4.1 berikut.

Tabel 4.1 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Sistem Operasi	Ubuntu 12.04 LTS
Bahasa Pemrograman	C, Python, Perl
Library	PocketSphinx versi 0.7, *SphinxBase* versi 0.7, *SphinxTrain* versi 1.07
Aplikasi Pendukung	GNOME Terminal 3.4.1.1, Gedit 3.4.1, LibreOffice Calc 3, Python 2.7.3, Perl 5, AN4,

Audacity *Voice Recoder*

4.1.2. Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam mengimplementasikan aplikasi pengenalan *voice command* berbahasa Indonesia ialah sebagai berikut:

Tabel 4.2 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
System Model PC	Toshiba Satellite L510
Processor	Pentium(R) Dual-Core CPU T4400 @ 2.20GHz × 2
Memory	2.0 GiB
Disk	17.3 GB
Os Type	32-bit
Kelengkapan	- 1 *Headphone Jack* - 1 *Microphone Input*

Penggunaan *microphone* berfungsi dalam perekaman data suara sebagai alat penunjang saat proses *training* dan *testing* program. Selain itu, penggunaan *microphone* juga membantu tingkat akurasi yang lebih baik pada saat melakukan kedua proses tersebut. Adapun spesifikasi teknis dari *microphone* ditunjukkan pada Tabel 5.2.

Tabel 4.3 Spesifikasi Teknis *Microphone*

Nama	Philips SBC MD150
Batas Frekuensi	*Headphone*: 8 Hz – 20KHz

Sensitivitas	*Microphone*: 8 Hz – 20KHz *Headphone*: 108 dB *Microphone*: -58 dB
Perangkat Tambahan	*Headphone*: 3.5 mm Stereo *Microphone*: 3.5 mm

4.2. Batasan-batasan Implementasi

Beberapa batasan implementasi dalam mengimplementasikan aplikasi pengenalan *voice command* Berbahasa Indonesia ialah sebagai berikut:

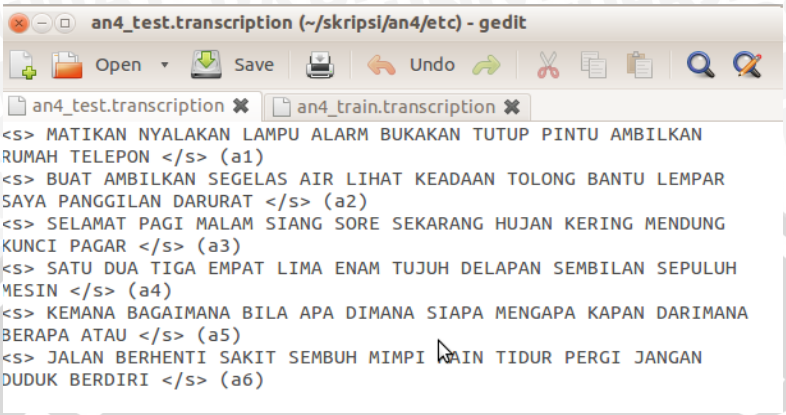
1. Implementasi yang dilakukan pada *voice command* ini hanya dapat mengubah dari suara manusia menjadi text (*speech to text*).
2. Perancangan ini dapat mengenali 25 kosakata Bahasa Indonesia.
3. Aplikasi ini menggunakan CMU Sphinx (PocketSphinx) sebagai *library* dalam perancangan ini.
4. Implementasi *aplikasi* pengenalan *voice command* berbahasa Indonesia menggunakan Bahasa pemrograman C.
5. Implementasi pengenalan kata bahasa Indonesia menggunakan metode *Hidden Markov Model*
6. Aplikasi dapat dijalankan tanpa menggunakan koneksi internet karena menggunakan konsep offline (local storage).
7. Interface aplikasi ini menggunakan *terminal* yang ada di ubuntu 12.04 LTS.

4.3. Implementasi *Training* Suara

Training suara dilakukan untuk mengenali ciri suara dari beberapa warna suara (manusia). *Training* suara dilakukan di sistem operasi *Ubuntu* 12.04. Ada beberapa *software* yang harus di-*install* terlebih dahulu, yaitu

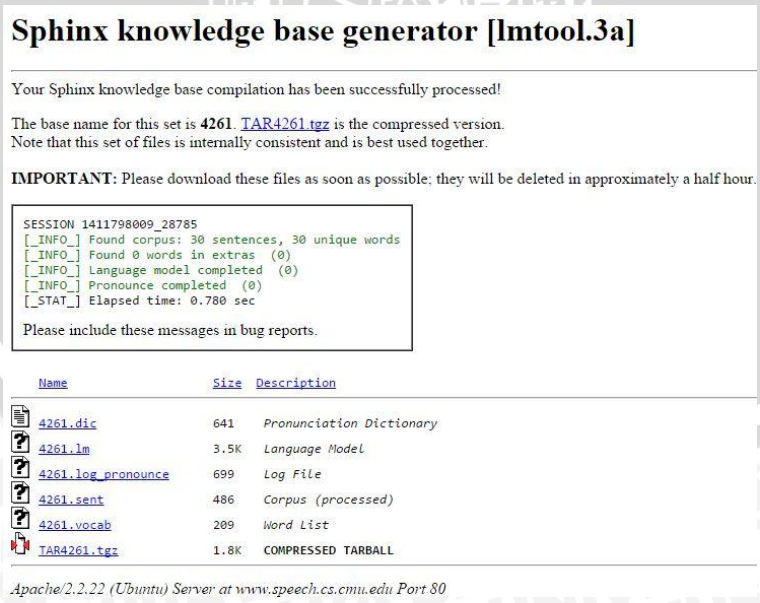
Fungsi dilakukannya training suara ialah agar hasil keluaran dari pelatihan dapat digunakan pada saat *testing* program. Dalam implementasi ini, langkah

pertama yang dilakukan ialah penulisan data artikel secara manual yang disesuaikan dengan kosakata pada penggunaan *smarthome system*. Selanjutnya ditulis pada teks editor yang ditampilkan dalam Gambar 4.1



Gambar 4.1 Bentuk Data Artikel

Selanjutnya data artikel disimpan dalam file *.txt* lalu diupload dengan URL: http://www.speech.cs.cmu.edu/tools/lmtool-new.html. Berdasarkan data yang diupload, lalu file upload tersebut akan menghasilkan model Bahasa, kamus pengucapan. Hasil ilustrasi file upload yang diolah lalu menghasilkan file-file penunjang aplikasi *voice command*, lalu diunduh keseluruhan file tersebut dan diolah secara manual isinya.



Gambar 4.2 Hasil Pengolahan Data Pengenalan Suara

Pengolahan *file-file* tersebut dilakukan secara manual yang disesuaikan

dengan pengucapan bahasa Indonesia. File yang pertama diolah ialah *dictionary file*. *Dictionary file* merupakan *file* yang berisi kamus fonetik. Kosakata pada kamus pengucapan ini harus disesuaikan dengan fonem suara orang Indonesia. contoh pengubahan kosakata dapat dilihat pada tabel 4.4.

Tabel 4.4 Contoh Transformasi Kosakata

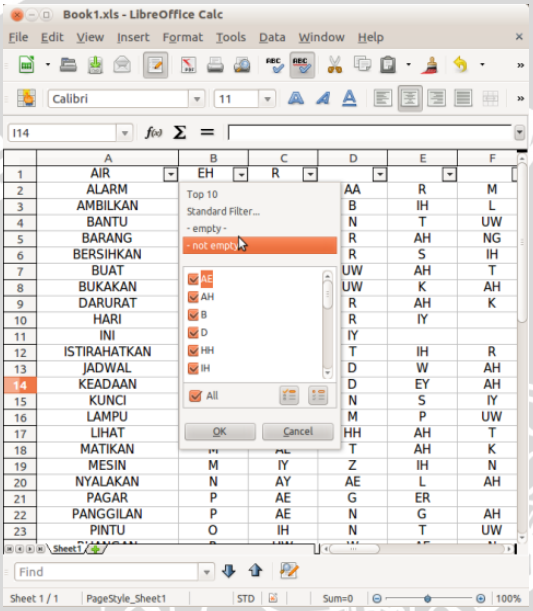
Pemisahan kosakata pada keadaan *default* (disamakan dengan pengucapan Bahasa Inggris)	
Air	EH ER
Pengeditan kosakata yang disesuaikan dengan suara orang Indonesia	
Air	AA IY IR

Penyesuaian pertama yang dilakukan ialah mengubah isi *dictionary file* yang isinya berupa kosakata dan pecahan-pecahan kosakata yang akan dikenali. Diusahakan pada saat pembuatan file ini disesuaikan dengan pengucapan orang Indonesia terhadap kata yang akan diuji. Pada gambar 4.3 merupakan penyesuaian kamus untuk pengenalan suara bahasa Indonesia.

```
1   AIR          AA IY R
2   ALARM        AA L AA R M
3   AMBILKAN     AA M B IY L K AA N
4   APA          AA P AA
5   ATAU         AA T AA UW
6   BAGAIMANA    B AA G EY M AA N AA
7   BANTU        B AA N T UW
8   BERAPA       B ER R AA P AA
9   BERDIRI      B ER R D IY R IY
10  BERHENTI     B ER R ER N T IY
11  BILA         B IH L AA
12  BUAT         B UW AA T
13  BUKAKAN      B UW K AA K AA N
14  DARIMANA     D AA R IH M AA N AA
15  DARURAT      D AE R UW R AA T
16  DELAPAN      D EH L AA P AA N
17  DIMANA       D IY M AA N AA
18  DUA          D UW AA
19  DUDUK        D UW D UW K
20  EMPAT        EH M P AA T
```

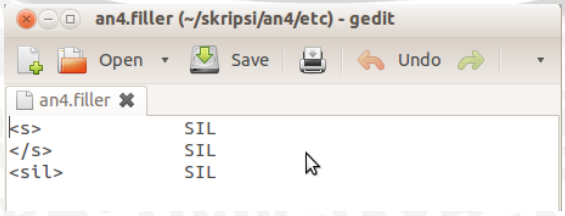
Gambar 4.3 Isi File Kamus Pengucapan

Berdasarkan isi *dictionary file*, implementasi yang dilakukan selanjutnya ialah membentuk file *phone* yang merupakan file yang berisi daftar fonetik yang digunakan dalam format ARPAbet. Pembuatan daftar fonetik menyesuaikan dengan pengucapan orang Indonesia terhadap kata yang akan diuji. Proses pembuatan *phone file* yakni data file *<nama_file>.phone* dimasukkan ke dalam *software* LibreOffice Calc yang dipisah dengan spasi. Penggambaran pemisahan file tersebut dengan *software* ditampilkan dalan Gambar 4.4.



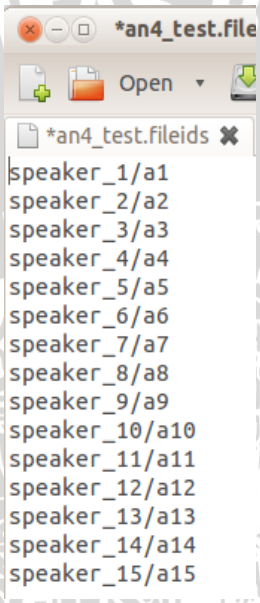
Gambar 4.4 Sortir Kata

Setelah membuat file kamus dan phone, peneliti mencoba membuat *file fillers*. *File filler* dapat dibuat secara manual berdasarkan rekaman suara pada keadaan sepi, sehingga penulisan "SIL" diartikan sebagai *silent* atau sepi. *File filler* juga memberikan tanda mulai dan berhenti kalimat pada *transcription file*. Adapun format penulisan secara default dari basis data suara di folder AN4. Bentuk *file filler* diilustrasikan dalam Gambar 4.5.



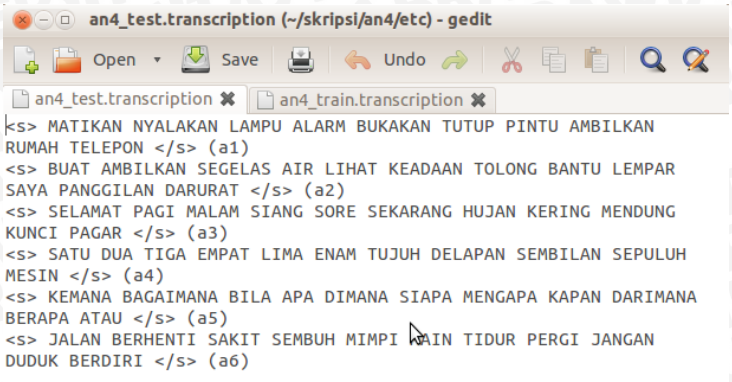
Gambar 4.5 Isi file SIL

Selanjutnya, ketika semua proses sebelumnya telah dilakukan, pengolahan manual yang dilakukan saat ini ialah membuat *file transcription*. *File transcription* berfungsi sebagai transkrip yang terdiri dari transkrip *training* berdasarkan kesesuaian file rekaman suaranya. Untuk membuat *file transcription* sesuai dengan peletakan file rekaman suara dibuatlah *file fileids* yang merupakan path/penunjuk file suara yang akan di*training*. Pada database peletakan *file* tersebut terdapat dua jenis *file transcription* dan *fileids* yaitu file *an4_train.fileids*, *an4_test.fileids* yang isi *file*nya disesuaikan satu sama lain, Gambar 4.6 menjabarkan kedua isi *file* tersebut.



Gambar 4.6 Isi File *an4_test.fileids* dan *an4_train.fileids*

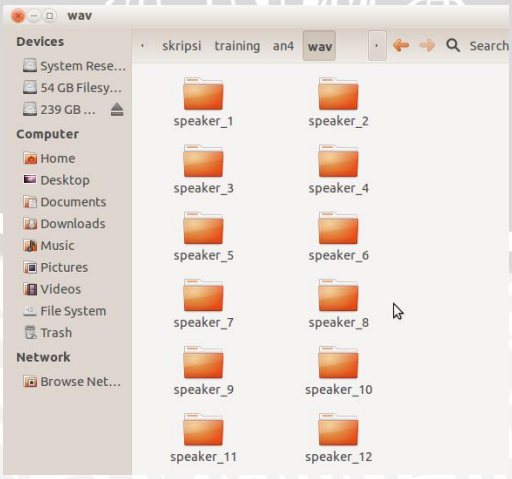
Sama halnya dengan *an4_train.transcription* dan *an4_test.transcription* yang berisi daftar semua kalimat yang telah dipasangkan dengan *file* suara untuk diuji coba, isi dari kedua *file* tersebut disesuaikan satu sama lain. Penjabaran isi kedua file tersebut dapat dilihat pada gambar 4.7.



Gambar 4.7 Isi File *an4_test.transcription* dan *an4_train.transcription*

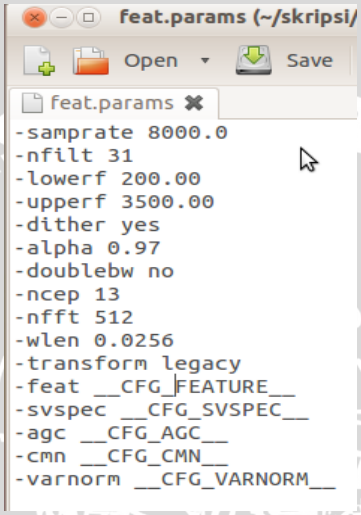
Setelah menyesuaikan *file transcription* dan *fileids* dibuatlah beberapa rekaman yang isi rekaman tersebut berdasarkan hasil *file transcription* yang sudah dibuat . Rekaman ini menggunakan 7 koresponden suara dengan perincian (3 laki-laki dan 4 perempuan). Hasil rekaman diletakkan pada folder *wav* yang ada pada AN4 dengan peletakan file sebagai berikut.

- *speaker*_1
  - o a1.wav
- *speaker*_2
  - o a2.wav
- *speaker*_3
  - o a3.wav
- dan seterusnya. Peletakan file suara *.wav* dapat dilihat pada gambar 4.8



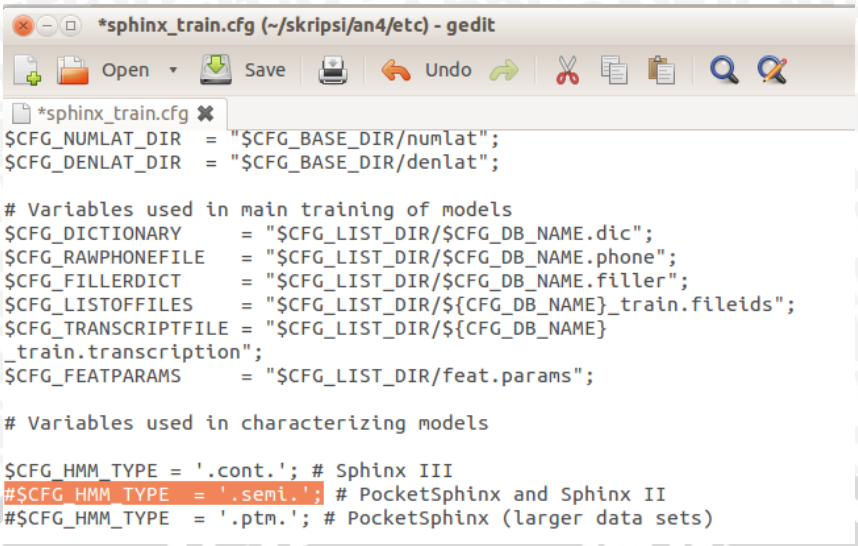
Gambar 4.8 Isi Folder Rekaman

Setelah folder rekaman diisi, selanjutnya merubah beberapa parameter yang digunakan untuk memaksimalkan hasil *training*. Pada saat training yang peneliti lakukan berdasarkan referensi yang peneliti dapat dari website CMU Sphinx, samprate menggunakan 8000 Hz, NFFT menggunakan 512, NFILT menggunakan 31 dan lowerf/ upperf menggunakan 200Hz & 3500Hz, penjabaran perubahan parameter dapat dilihat pada gambar 4.9.



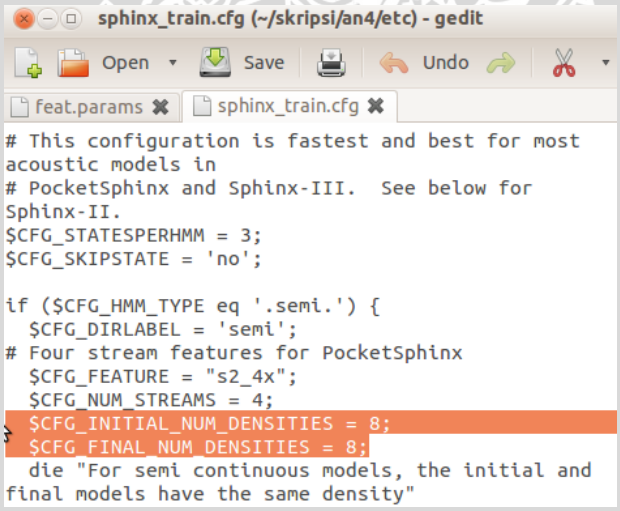
Gambar 4.9 Isi File *feat.params*

Berdasarkan tutorial yang diberikan oleh website CMU Sphinx untuk pembuatan *training acoustic model*, terdapat 3 tipe model jenis perekaman, yakni *continuous* (*cont*), *semi-continuous* (*semi*), dan *phonetically tied mixtures* (*ptm*). Model *ptm* digunakan jika jumlah data perekaman yang cukup banyak. Model *semi* digunakan jika jumlah data perekaman sedikit dan membutuhkan kecepatan akses saat proses *testing*. Model *cont* digunakan jika jumlah data perekaman berada di antara model *ptm* dan *semi*. Pada implementasi saat ini digunakan model *semi-continuous* model seperti dalam Gambar 4.10.



Gambar 4.10 Model Tipe Semi

Parameter yang digunakan terdiri dari *density* dan *senone*. Jumlah *density* tergantung pada jumlah kosakata yang dimiliki basis data. Penulisan *density* berkelipatan 2, seperti 1, 4, 8, 16, 32, 64. Penjabaran penulisan *density* dalam Gambar 4.11.



Gambar 4.11 Besar *Density*

Penulisan *senone* juga berdasarkan jumlah kosakata, dan waktu *training* kosakata sehingga diperlukan nilai yang optimal. Pada basis data Bahasa Indonesia digunakan jumlah kosakata, waktu *training* dan *density* yang minimal. Jika penulisan *senone* terlalu banyak, muncul pesan kesalahan sewaktu proses *training* dalam *folder log*. Penulisan *senone* diilustrasikan dalam Gambar 4.30.

```
# (yes/no) Train multiple-gaussian context-independent models
(useful
# for alignment, use 'no' otherwise) in the models created
# specifically for forced alignment
$CFG_FALIGN_CI_MGAU = 'no';
# (yes/no) Train multiple-gaussian context-independent models
(useful
# for alignment, use 'no' otherwise)
$CFG_CI_MGAU = 'no';
# Number of tied states (senones) to create in decision-tree
clustering
$CFG_N_TIED_STATES = 200;
# How many parts to run Forward-Backward estimatinon in
$CFG_NPART = 1;
```

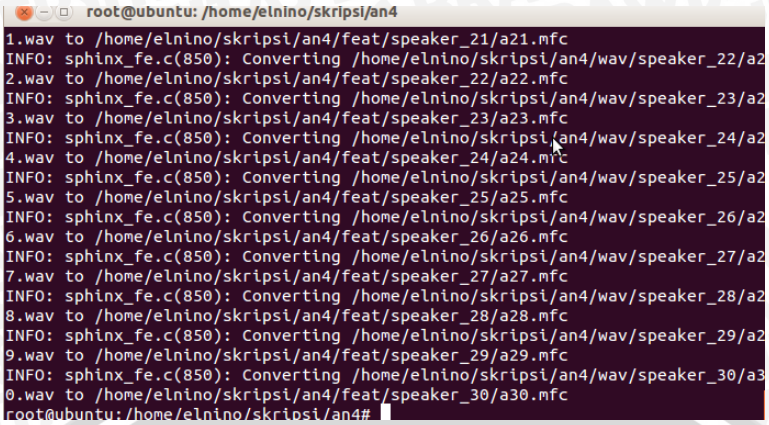
Gambar 4.12 Jumlah *Tiedstate*

Berikut ini perkiraan perbandingan penggunaan *senones* dan *densities* untuk pelatihan *training* suara. Pada penelitian kali ini saya menggunakan senones "200" dan density "8". Penggunaan nilai *senones* dan *density* tersebut berdasarkan banyaknya basis data rekaman yang sudah saya lakukan, apabila menggunakan vocabulary minimal maka digunakan nilai yang minimal juga. Peraturan penggunakan *senones* dan *density* dilihat pada tabel 4.4.

Tabel 4.5 Perbandingan Nilai *senones* dan *densities* (Sumber: Training Acoustic Model CMU Sphinx Documentation)

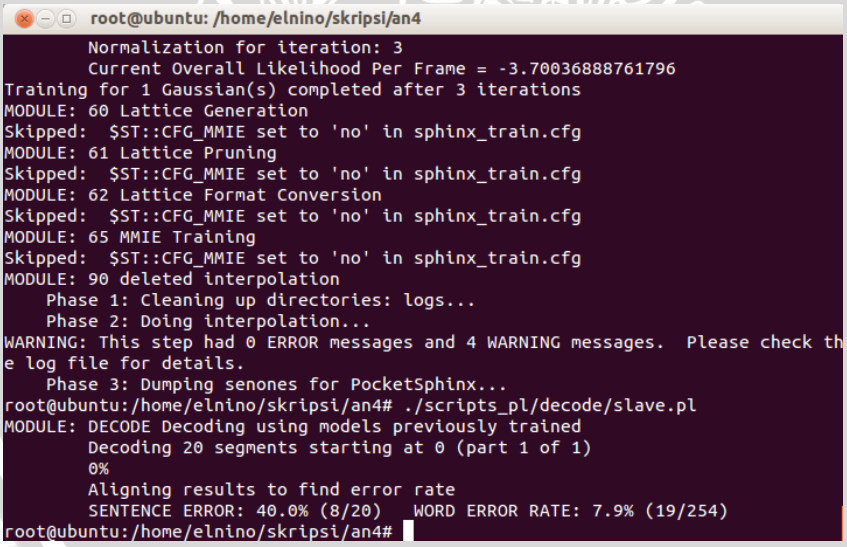
Vocabulary	Hours in db	Senones	Densities	Example
20	5	200	8	Tidigits Digits Recognition
100	20	2000	8	RM1 Command and Control
5000	30	4000	16	WSJ1 5k Small Dictation
20000	80	4000	32	WSJ1 20k Big Dictation
60000	200	6000	16	HUB4 Broadcast News
60000	2000	12000	64	Fisher Rich Telephone Transcription

Setelah melakukan konfigurasi terhadap nilai-nilai untuk persiapan training, ketikkan "*./scripts_pl/make_feats.pl –ctl etc/an4_train.fileids*" untuk melakukan mengkonversi file suara *wav* menjadi *mfc* dalam bentuk *binary*. Hasil konversi *file* dapat dilihat pada gambar 4.13



Gambar 4.13 Proses Pembentukan File *.mfc*

Pada sintaks yang dilakukan pada gambar 4.13 didapatkan pembentukan file baru yang digunakan untuk melakukan proses *training* suara yang dilakukan oleh *library* SphinxTrain Setelah dilakukan tahap konfigurasi basis data suara, dilanjutkan dengan penggunaan *library* SphinxTrain pada tahap *training* file suara suara telah dipetakan pada *transcription file* dan telah disesuaikan dengan *file* rekaman pada *folder wav* dengan perintah *./scripts.pl/RunAll.pl*.



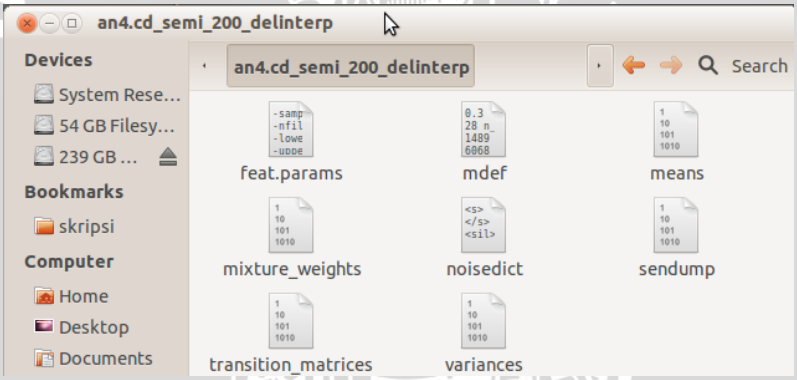
Gambar 4. 14 Modul 90. *deleted_interpolation* & Modul *Decode*

Pada gambar 4.14 hasil keseluruhan proses *training* suara untuk 254 kosakata memiliki akurasi sebesar 94.88%, kata-kata yang berada dalam *file* audio dapat dikenali berupa kata yang sama dengan *file transcription* seperti dalam Gambar 4.7. dan hasil lebih lengkap pada **Lampiran 2**.

```
LAMPU rumah segelas AIR tolong lihat keadaan tolong bantu lempar
saya berhenti sembuh tidur(SPEAKER_19-A19)
Words: 14 Correct: 12 Errors: 2 Percent correct = 85.71% Error =
14.29% Accuracy = 85.71%
Insertions: 0 Deletions: 0 Substitutions: 2
segelas sekarang selamat sembilan sembuh sepuluh siang siapa sore
telepon tidur tiga matikan mesin  (SPEAKER_20-A20)
segelas sekarang selamat sembilan sembuh sepuluh siang siapa sore
telepon tidur tiga matikan mesin  (SPEAKER_20-A20)
Words: 14 Correct: 14 Errors: 0 Percent correct = 100.00% Error =
0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
TOTAL Words: 254 Correct: 241 Errors: 13
TOTAL Percent correct = 94.88% Error = 5.12% Accuracy = 94.88%
TOTAL Insertions: 0 Deletions: 3 Substitutions: 10
```

Gambar 4.15 Hasil Proses *Training*

Hasil proses *training* ini menghasilkan *Acoustic Model* yang dapat memaksimalkan pada saat melakukan *testing* pengenalan suara. *Acoustic Model* dapat diambil dari folder *model_parameters* seperti dalam Gambar 4.16.



Gambar 4.16 *Acoustic Model*

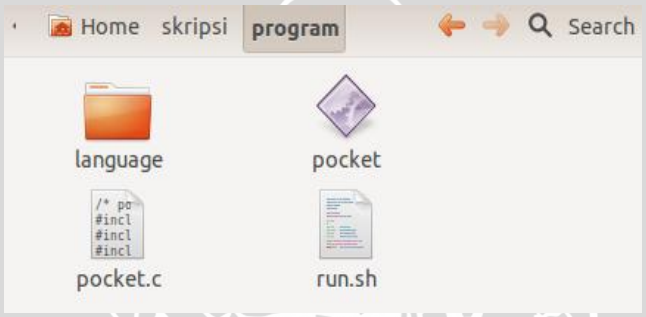
Training dapat dilakukan secara bertahap yang terdiri beberapa modul yang ada pada folder SphinxTrain. Pada setiap modulnya memiliki beberapa fungsi yang membantu pada saat proses *training*. Pada tabel 4.6 menunjukan penjabaran beberapa proses *training* berdasarkan pembagian modulnya. *Training* membantu untuk membangkitkan probabilitas suatu kosakata, apabila kosakata sering dilatih maka akurasi kosakata yang diucapkan pada saat *testing* akan semakin baik.

Tabel 4.6 Penjelasan Modul *Training*

Nama Modul	Fungsi
000.*comp_feat*	Membuat direktori baru yang terdiri dari beberapa *file* kosong, dan sinyal suara diubah menjadi vektor fitur MFCC yang diubah ke dalam direktori *feat*.
00.*verify*	Pengecekan dan penyesuaian *file* yang dibutuhkan selama proses *training*. Proses *training* tidak dapat dijalankan ketika kelengkapan dari file suara, kamus, transkrip dan file path tidak sesuai.
20.*ci_hmm*	Melakukan proses *training* model *Context-Independent* (CI) pada setiap kata dalam *file dictionary*
30.*cd_hmm_untied*	Melakukan pelatihan model *Context-Dependent* dengan *untied-state* (CD-untied) berdasarkan model *Context-Independent* (CI) pada setiap kata dalam kamus.
40.*buildtrees*	Pembangunan *decision tree* untuk setiap *state* dari tiap kosakata.
45.*prunetree*	Membuat pemangkasan *decision tree* dan *tied-state*
50.*cd_hmm_tied*	Pelatihan model *CD-tied* menggunakan HMM
90. *deleted_interpolation*	Merupakan teknik N-gram *smoothing*
Decode	Pengambilan model, dilakukan pengujian berdasarkan rekaman suara dengan *file transcription* untuk estimasi kualitas model berdasarkan *Word Error Rate* (WER) dan *Sentence Error*.

4.4. Implementasi Testing Program

Implementasi ini merupakan hasil dari implementasi training suara. Setelah melakukan training suara, secara otomatis akan menghasilkan beberapa file yang dapat digunakan untuk mencoba program diantaranya kamus pengucapan, model Bahasa dan model akustik yang didapatkan dari proses training suara, lalu ditempatkan pada *folder* yang sama dengan programnya. Untuk menempatkan file model akustik, pilihlah *folder* "model_parameters" yang berada pada file AN4, lalu pilihlah *folder* "an4.cd_semi 200" yang isi filenya akan digunakan sebagai isi dari *folder* "hmm" pada *library* PocketSphinx. (Untuk hasil pengambilan datanya dapat dilihat pada gambar 4.16).



Gambar 4.17 Isi Folder Program

Dalam pengujian testing suara, adanya kebutuhan beberapa file seperti *language model*, *acoustic model*. Selanjutnya dari beberapa kebutuhan file tersebut maka dapat dipanggil seperti *listing code* dibawah ini:

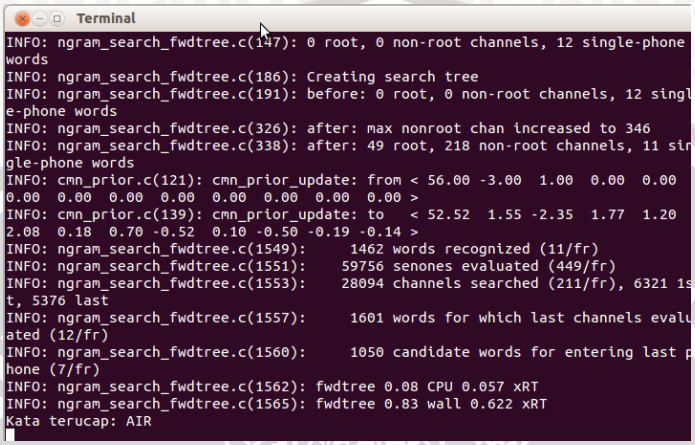
```
int main (int argc, char *argv[])
{

        if (initElements (
                        MODELDIR "/0389.lm",
                        MODELDIR "/0389.dic",
                        NATIVE_MODELDIR "/hmm/en_US/hub4wsj_sc_8k" ))
                play ();
        else
        {
                printf("Init failed...");
                return -1;
        }

        return 0;
}
```

Gambar 4.18 Pemanggilan Data

Selanjutnya pengucapan kata dengan *microphone*. Sinyal suara dari *microphone* diekstraksi fiturnya dan dilakukan proses dekoding secara otomatis dengan *library* PocketSphinx sehingga kata yang diucapkan ialah “air” lalu dapat dihasilkan, seperti dalam Gambar 4.19.

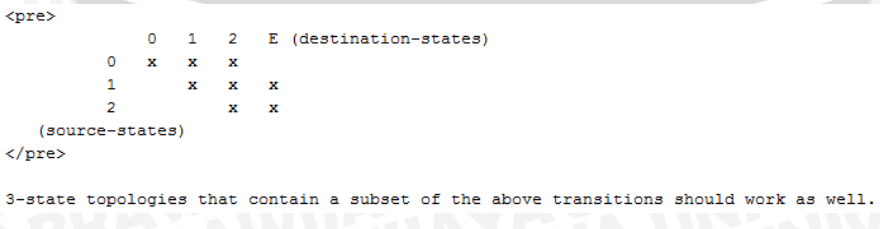


Gambar 4.19 Hasil Output Program

4.5 Implementasi Metode Hidden Markov Model terhadap program

Penggunaan Metode Hidden Markov Model pada pengenalan suara sangat popular digunakan karena dapat menunjang pelatihan yang sederhana dan otomatis pada aplikasi serta mudah dalam komputasinya. Pada sub-bab ini akan menerangkan bagaiman kinerja Metode HMM pada aplikasi *voice command speech to text*.

Hidden Markov model pada *library* PocketSphinx digunakan sebagai penghitung nilai terbaik berdasarkan probabilitas akustik dari sebuah pengucapan. Perhitungan probabilitas tersebut didapat berdasarkan *state-state*. Biasanya HMM menggunakan tiga state sebagai dasar perhitungan probabilitasnya.



Gambar 4.20 Struktur 3 State HMM

Penggunaan HMM pada PocketSphinx dapat dilihat di folder PocketSphinx di sub-folder library (PocketSphinx/src/lib/hmm.c). Disana terdapat elemen-elemen yang digunakan sebagai perhitungan untuk mendapatkan probabilitas terbaik. Salah satunya terdapat pada listing code dibawah ini.

```
s1 = hmm_score(hmm, 1) + nonmpx_senscr(1);
/* All transitions into state 3 */
if (s1 BETTER_THAN WORST_SCORE) {
    t0 = s3 + hmm_tprob_5st(3, 3);
    t1 = s2 + hmm_tprob_5st(2, 3);
    t2 = s1 + hmm_tprob_5st(1, 3);
    if (t0 BETTER_THAN t1) {
        if (t2 BETTER_THAN t0) {
            s3 = t2;
            hmm_history(hmm, 3)  = hmm_history(hmm, 1);
        } else
            s3 = t0;
    } else {
        if (t2 BETTER_THAN t1) {
            s3 = t2;
            hmm_history(hmm, 3)  = hmm_history(hmm, 1);
        } else {
            s3 = t1;
            hmm_history(hmm, 3)  = hmm_history(hmm, 2);
        }
    }
    if (s3 WORSE_THAN WORST_SCORE) s3 = WORST_SCORE;
    if (s3 BETTER_THAN bestScore) bestScore = s3;
    hmm_score(hmm, 3) = s3;
}
```

Gambar 4.21 Probabilitas HMM

Disimpulkan berdasarkan *listing code* diatas yang diambil pada saat perhitungan probabilitas pada state-1. Perbandingan tersebut dilihat dari *source state* dan *destination state* (lihat gambar 4.20). Perhitungan lalu dibandingkan dengan state-state yang lain, dan dilihat nilai terbaik dari masing-masing state. Sehingga HMM memunculkan probabilitas terbaik pada saat manusia mengucapkan suaranya, lalu diterjemahkan oleh kamus pengucapan, model Bahasa dan akustik model. Dari masing-masing state-state yang saling memberikan informasi sehingga ketika tidak ditemukan parameter yang dikehendaki, HMM melihat itu sebagai parameter yang tersembunyi dan mencari kemungkinan yang hampir sama dengan parameter yang dikehendaki. Gambar 4.22 memperlihatkan ketika hasil probabilitas yang didapat kecil berdasarkan hasil perhitungan probabilitas yang lama, maka akan diambil berdasarkan hasil probabilitas yang tertinggi.

```
/* All transitions into state 1 */
t0 = s1 + hmm_tprob_5st(1, 1);
t1 = s0 + hmm_tprob_5st(0, 1);
if (t0 BETTER_THAN t1) {
    s1 = t0;
} else {
    s1 = t1;
    hmm_history(hmm, 1)  = hmm_in_history(hmm);
}
if (s1 WORSE_THAN WORST_SCORE) s1 = WORST_SCORE;
if (s1 BETTER_THAN bestScore) bestScore = s1;
hmm_score(hmm, 1) = s1;

/* All transitions into state 0 */
s0 = s0 + hmm_tprob_5st(0, 0);
if (s0 WORSE_THAN WORST_SCORE) s0 = WORST_SCORE;
if (s0 BETTER_THAN bestScore) bestScore = s0;
hmm_in_score(hmm) = s0;

hmm_bestscore(hmm) = bestScore;
return bestScore;
}
```

Gambar 4.22 Perbandingan Nilai Terbaik untuk Beberapa State

```
MODULE: 20 Training Context Independent models
   Phase 1: Cleaning up directories:
       accumulator...logs...qmanager...models...
   Phase 2: Flat initialize
   Phase 3: Forward-Backward
       Baum welch starting for 1 Gaussian(s), iteration: 1 (1 of 1)
       0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
       Normalization for iteration: 1
       Current Overall Likelihood Per Frame = -14.1096482699457
       Baum welch starting for 1 Gaussian(s), iteration: 2 (1 of 1)
       0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
       Normalization for iteration: 2
       Current Overall Likelihood Per Frame = -7.33229625393194
       Convergence Ratio = 6.77735201601376
       Baum welch starting for 1 Gaussian(s), iteration: 3 (1 of 1)
       0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
```

Gambar 4.23 Proses HMM pada saat training

Berdasarkan hasil implementasi HMM pada library PocketSphinx yang mendasari teori-teori yang dibutuhkan pada aplikasi pengenalan suara, selain untuk mendapatkan hasil terbaik, metode ini bersifat stokastik yang artinya apabila diberikan inputan keadaan saat ini, keadaan akan datang dapat diprediksi dan ia lepas dari keadaan di masa lampau. Deskripsi kondisi saat ini menangkap semua informasi yang mempengaruhi evolusi dari suatu sistem di masa depan.

BAB V
PENGUJIAN DAN ANALISIS

Pada bab ini dibahas mengenai pengujian dan analisis sistem. Pengujian yang dilakukan terdiri dari 2 macam, yakni pengujian *training* suara untuk menentukan parameter terbaik berdasarkan *word error rate* dan pengujian validitas (keakuratan) program dalam mengenali Bahasa Indonesia. Tujuan dari pengujian yakni penilaian kesesuaian antara implementasi program dan analisis kebutuhan. Tujuan dari analisis yakni pemberian kesimpulan dari hasil pengujian.

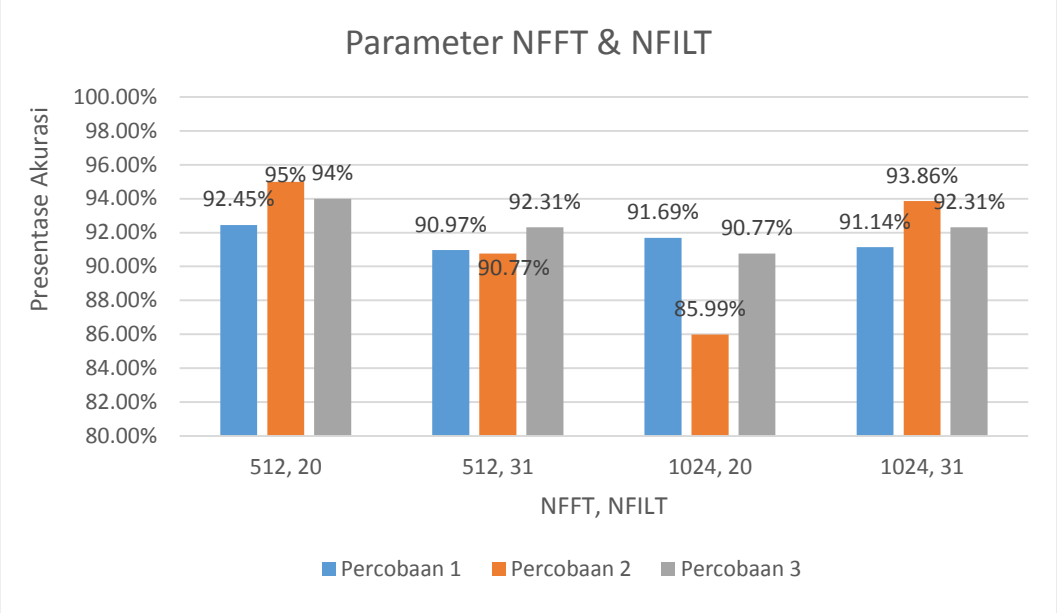
5.1. Pengujian *Training* Suara

Pengujian ini bertujuan untuk mengurangi kata yang *error* berdasarkan *training* suara yang dilakukan berdasarkan perubahan parameter. Perubahan parameter sangat berdampak terhadap optimasi penggunaan aplikasi. Proses *training acoustic model* ini menggunakan *library* SphinxTrain. Pada pengujian ini user akan mengubah beberapa nilai parameter yang dapat menghasilkan hasil keluaran yang lebih baik. Perubahan parameter ini didapat dari pengujian *training* terhadap WER (Word Error Rate). Pengujian WER (Word Error Rate) ini merupakan pengukuran terhadap kata-kata yang eror pada saat *training*. Hasil ini didapat dari sintak yang sebelumnya dijalankan pada bab implementasi sehingga menghasilkan *folder result* yang ada di AN4 kemudian dapat dilihat error dan presentase WER. Hasil pengujian WER dapat dilihat di **Lampiran 2**.

5.1.1. Parameter NFFT dan NFILT

Parameter NFFT dan NFILT merupakan parameter untuk mengurangi *frame eror* pada saat pelatihan terhadap perekaman. Dengan merubah nilai parameter NFFT dan NFILT ini, akurasi yang dihasilkan dapat berbeda-beda. Berdasarkan tutorial *training* suara *acoustic model* yang dikeluarkan oleh CMU Sphinx nilai terbaik pada perekaman yang menggunakan *samprate* 8 *khz* ialah NFFT (512 dan 1024) dan NFILT (20 dan 31). Dari semua perubahan nilai parameter tersebut lalu dikombinasikan untuk mendapatkan hasil keakuratan yang

paling baik.Untuk hasil pengujian lengkap dapat dilihat pada grafik pada gambar 5.1



Gambar 5.1 Grafik Pengujian NFFT dan NFILT

Pada parameter nfft=512 dan nfilt=20 dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 235 sehingga menghasilkan akurasi sebesar 92.45%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak 241 sehingga menghasilkan akurasi sebesar 95%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 241 sehingga menghasilkan akurasi sebesar 95%.

Pada parameter nfft=512 dan nfilt=31 dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 231 sehingga menghasilkan akurasi sebesar 90.97%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak 230 sehingga menghasilkan akurasi sebesar 90.77%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 234 sehingga menghasilkan akurasi sebesar 92.31%.

Pada parameter nfft=1024 dan nfilt=20 dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 232 sehingga menghasilkan akurasi sebesar 91.69%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak 218 sehingga

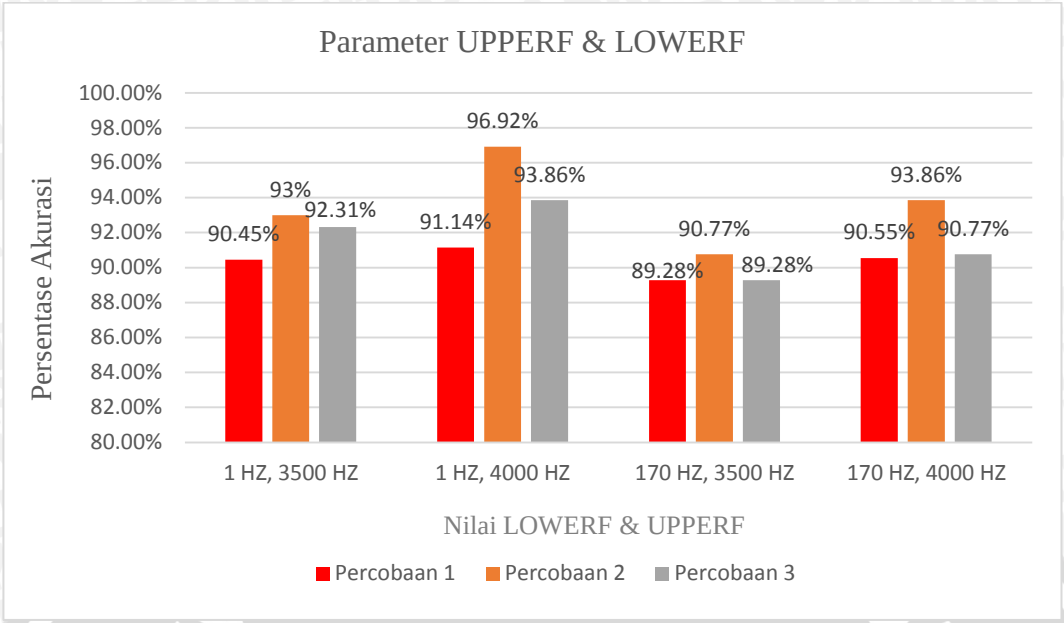
menghasilkan akurasi sebesar 85.99%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 230 sehingga menghasilkan akurasi sebesar 90.77%.

Pada parameter nfft=1024 dan nfilt=31 dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 232 sehingga menghasilkan akurasi sebesar 91.14%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak 238 sehingga menghasilkan akurasi sebesar 93.86%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 234 sehingga menghasilkan akurasi sebesar 92.31%.

Berdasarkan hasil pengujian yang dilakukan sebanyak 3 kali perekaman suara berdasarkan 4 koresponden suara yang berbeda maka didapat nilai optimum berdasarkan hasil pada gambar 5.1 untuk parameter nfft dan nfilt yakni nfft = 512 dan nfilt= 20. Maka dengan pengujian ini nilai parameter nfft dan nfilt terbaik akan digunakan sebagai pedoman untuk pengujian *testing* program *speech to text*. Hasil akurasi pada gambar 5.1 merupakan hasil *training* yang didapat berdasarkan pengubahan nilai nfft dan nfilt. Uraian hasil akurasi dapat dilihat pada **Lampiran 2.** Hasil akurasi yang dilampirkan pada *training* merupakan hasil akurasi terbaik berdasarkan perubahan parameternya.

5.1.2. Parameter Lowerf dan Upperf

Parameter Lowerf dan Upperf digunakan untuk membatasi nilai bawah dan atas terhadap frekuensi suara yang digunakan. Akurasi yang dihasilkan juga dipengaruhi dengan merubah nilai parameter lowerf dan upperf ini. Berdasarkan tutorial *training* suara *acoustic model* yang dikeluarkan oleh CMU Sphinx nilai terbaik pada perekaman yang menggunakan *samprate* 8 *khz* ialah Lowerf (1 Hz, 150 Hz) dan Upperf (3500 Hz, 4000 Hz) Untuk hasil pengujian lengkap dapat dilihat pada gambar 5.2.



Gambar 5.2 Grafik Pengujian Parameter Lowerf & Upperf

Pada parameter Upperf=1 Hz dan Lowerf=3500 Hz dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 229 sehingga menghasilkan akurasi sebesar 90.45%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak 236 sehingga menghasilkan akurasi sebesar 93%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 234 sehingga menghasilkan akurasi sebesar 92.31%.

Pada parameter Upperf=1 Hz dan Lowerf=4000 Hz dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 232 sehingga menghasilkan akurasi sebesar 91.14%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak 246 sehingga menghasilkan akurasi sebesar 96.92%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 238 sehingga menghasilkan akurasi sebesar 93.86%.

Pada parameter Upperf=170 Hz dan Lowerf=3500 Hz dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 227 sehingga menghasilkan akurasi sebesar 89.28%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak

230 sehingga menghasilkan akurasi sebesar 90.77%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 227 sehingga menghasilkan akurasi sebesar 89.28%.

Pada parameter Upperf=170 Hz dan Lowerf=4000 Hz dilakukan pengujian sebanyak 20 kalimat yang terdiri dari 254 kosakata, pada percobaan pertama jumlah kosakata yang dikenali sebanyak 229 sehingga menghasilkan akurasi sebesar 90.55%, pada percobaan kedua jumlah kosakata yang dikenali sebanyak 238 sehingga menghasilkan akurasi sebesar 93.86%, pada percobaan ketiga jumlah kosakata yang dikenali sebanyak 230 sehingga menghasilkan akurasi sebesar 90.77%.

Berdasarkan hasil pengujian yang dilakukan sebanyak 3 kali perekaman suara peneliti sendiri maka didapat nilai optimum berdasarkan grafik pada gambar 5.2 untuk parameter lower frekuensi dan upper frekuensi yakni Lowerf = 1 Hz dan Upperf = 4000 Hz. Akurasi yang dihasilkan untuk setiap database yang digunakan oleh nilai parameter ini merupakan akurasi tertinggi. Sehingga *word error rate* untuk sistem dapat ditekan sekecil mungkin agar pencarian kata menggunakan ucapan dapat lebih baik.

Hasil pengujian ini dilakukan untuk mengetahui rekaman suara yang dilatih pada sistem dapat dikenali dengan baik atau tidak. Disimpulkan bahwa kombinasi dari nilai NFFT, NFILT, Lowerf dan Upperf tertentu menghasilkan word error rate yang berbeda-beda. Maka didapati kesimpulan bahwa nilai parameter dengan Nfft = 512, Nfilt = 20, Lowerf = 1 Hz dan Upperf = 4000Hz merupakan nilai optimum untuk mengimplementasikan *voice command speech to text* menggunakan PocketSphinx.

5.2. Pengujian Validasi

Dalam pengujian validasi dituliskan daftar analisis kebutuhan sebagai acuan selama pengujian. Setelah dilakukan pengujian, dapat dibandingkan antara analisis kebutuhan dan hasil pengujian validasi dinilai berdasarkan persentase keakuratan dalam mengenali suara bahasa Indonesia.

5.2.1. Pengujian Koresponden Suara

Pengujian kali ini dilakukan 2 tahap, yakni pengujian keakuratan menggunakan 5 koresponden suara dengan mengucapkan 25 kata dengan banyaknya perulangan ucapan sebanyak 3 kali serta pengujian keakuratan menggunakan 5 koresponden suara dengan menguji salah satu kata yang diucapkan berulang sebanyak 50 kali. Tujuan dilakukan pengujian ini untuk menentukan kinerja program terhadap suara yang berbeda. Berikut hasil pengujiannya dijabarkan oleh tabel 5.1 dan untuk pengucapan sebanyak 50 kali dapat dilihat pada tabel 5.2 serta pada tabel 5.3 yang menjabarkan hasil perincian pengucapan sebanyak 50 kali tersebut.

Tabel 5.1 *Testing* Koresponden Suara

No	Kata	Koresponden 1		Koresponden 2		Koresponden 3		Koresponden 4		Koresponden 5		Akurasi Per-kata
		Banyak Pengulangan	Banyak dikenali	Banyak Pengulangan	Banyak dikenali	Banyak Pengulangan	Banyak dikenali	Banyak Pengulangan	Banyak dikenali	Banyak Pengulangan	Banyak dikenali	
1	Air	3	3	3	3	3	3	3	3	3	1	86.7%
2	Alarm	3	3	3	0	3	3	3	0	3	3	60%
3	Ambilkan	3	3	3	3	3	2	3	3	3	3	98.38%
4	Bantu	3	3	3	2	3	3	3	3	3	1	80%
5	Berdiri	3	2	3	3	3	3	3	3	3	1	80%
6	Berhenti	3	3	3	3	3	3	3	0	3	0	60%
7	Darurat	3	2	3	3	3	1	3	3	3	2	80.6%
8	Duduk	3	3	3	3	3	2	3	3	3	3	93.3%
9	Jalan	3	3	3	3	3	0	3	2	3	0	70.9%
10	Jangan	3	3	3	2	3	0	3	0	3	0	67.7%
11	Keadaan	3	3	3	3	3	3	3	3	3	2	93.3%
12	Kunci	3	3	3	1	3	3	3	3	3	2	80%
13	Lampu	3	3	3	3	3	3	3	1	3	3	86.67%
14	Lihat	3	3	3	3	3	3	3	3	3	2	93.3%
15	Main	3	3	3	3	3	3	3	3	3	2	93.3%

16	Matikan	3	3	3	2	3	3	3	3	3	3	93.3%
17	Pagar	3	3	3	3	3	3	3	1	3	2	80%
18	Pintu	3	3	3	3	3	3	3	1	3	3	86.67%
19	Rumah	3	3	3	3	3	3	3	3	3	2	93.3%
20	Sakit	3	3	3	3	3	3	3	3	3	3	100%
21	Satu	3	3	3	0	3	3	3	3	3	3	80%
22	Segelas	3	3	3	0	3	3	3	3	3	3	80%
23	Telepon	3	3	3	3	3	3	3	3	3	3	100%
24	Tolong	3	3	3	1	3	1	3	0	3	2	46.67%
25	Tutup	3	0	3	3	3	3	3	3	3	2	77.4%
Akurasi Per-Koresponden		92%		78%		84%		76%		74%		
Total Akurasi Per-Kata		**83%**										
Total Akurasi Per-Koresponden		**80.8%**										

Tabel 5.2 Pengucapan Sebanyak 50 kali

No. Koresponden	Kata	Koresponden 1		Akurasi Per-kata
		Banyak Pengulangan	Banyak dikenali	
1	Ambilkan	50	50	100%
2	Jangan	50	39	78%
3	Darurat	50	40	80%
4	Duduk	50	38	76%
5	Tutup	50	40	80%

Tabel 5.3 Perincian *Testing* Kata Sebanyak 50 kali

Koresponden Suara	Kata yang diucapkan	Kata Output	Akurasi
Pertama	Ambilkan	Ambilkan, Ambilkan,	100%
Kedua	Jangan	Jangan, Jalan, Jangan, Jalan, Jangan, Jangan, Jangan, Jangan, Jangan, Jangan, ****, Jangan, Jangan, Jangan, Jangan, Jalan, Jangan, Jangan, ****, Jangan, Jangan, Jangan, Jangan, Jangan, Jalan, Jangan, Jangan, ****, Jangan, Jangan, Jangan, Jangan, Jalan, Jangan, Jangan, Jangan, Jangan, Jangan, Jangan, Jalan, Jangan, Jalan, Jangan, Jangan, Jangan, Jangan, ****, Jangan,	78%

		Jangan, Jangan,	
Ketiga	Darurat	Darurat, Darurat, Lihat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Lihat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Lihat, Lihat, Darurat, Darurat, Darurat, Darurat, Darurat, Darurat, Lihat, Darurat, Lihat, Darurat, Darurat, Darurat, Lihat, Darurat, Lihat, Darurat, Darurat, Darurat, Darurat, Lihat, Darurat, Lihat, Darurat, Darurat,Darurat	80%
Keempat	Jalan	Jalan, Jalan, Jalan, Jalan, Jangan, Jalan, Jalan, ****, Jalan, Jalan, Jangan, Jalan, Jalan, Jalan, Jalan, Jalan, ****, Jalan, Jalan, Jalan, Jalan, Jalan, Jalan, Jangan, Jalan, Jalan, Jalan, ****, Jalan, Jalan, Jalan, Keadaan, Jalan, Jalan, Jalan, Jangan, Jalan, Jalan, Jalan, Jangan, Jalan, Jalan, Jalan, Jalan, ****, Jalan, Keadaan, Jalan, Jangan, Jalan,	76%
Kelima	Tutup	Tutup, Tutup, Tutup, Pintu, Tutup, Tutup, Tutup, Tutup,Tutup,Pintu,Tutup, Pintu, Tutup, Tutup, Tutup, Tutup,****, Tutup, Tutup, Tutup, Tutup,Pintu, Tutup, Tutup,Tutup, Satu, Tutup, Tutup, Tutup, Tutup, Telepon, Tutup, Tutup, Tutup, Tutup,Pintu, Tutup, Tutup, Tutup, Tutup,****, Tutup, Tutup, Tutup, ****, Tutup, Tutup	80%

*Keterangan : ****** (Kata tidak dikenali)*

5.2.2. Pengujian Waktu

Pada tabel 5.4 akan dilampirkan perincian akurasi pengucapan kosakata terhadap waktu. Pengujian terhadap waktu bertujuan mengetahui keakuratan waktu sehingga nantinya mudah diterapkan pada *smarthome system*. Keadaan waktu yang penting diketahui untuk menunjang perancangan *smarthome system* yang harus bersifat *real-time*.

Tabel 5.4 Pengujian Kosakata Terhadap Waktu

No	Kata	Pengujian			Rata-rata waktu (ms)
		Pengujian I	Pengujian II	Pengujian III	
		Akurasi Waktu (ms)	Akurasi Waktu (ms)	Akurasi Waktu (ms)	
1	Air	410	500	510	473.33
2	Alarm	510	410	310	410
3	Ambilkan	500	510	410	473.33
4	Bantu	450	450	410	436.67
5	Berdiri	410	410	410	410
6	Berhenti	410	410	610	476.67
7	Darurat	510	510	410	476.67
8	Duduk	410	410	410	410
9	Jalan	510	510	410	476.67
10	Jangan	610	410	410	476.67
11	Keadaan	610	410	410	476.67

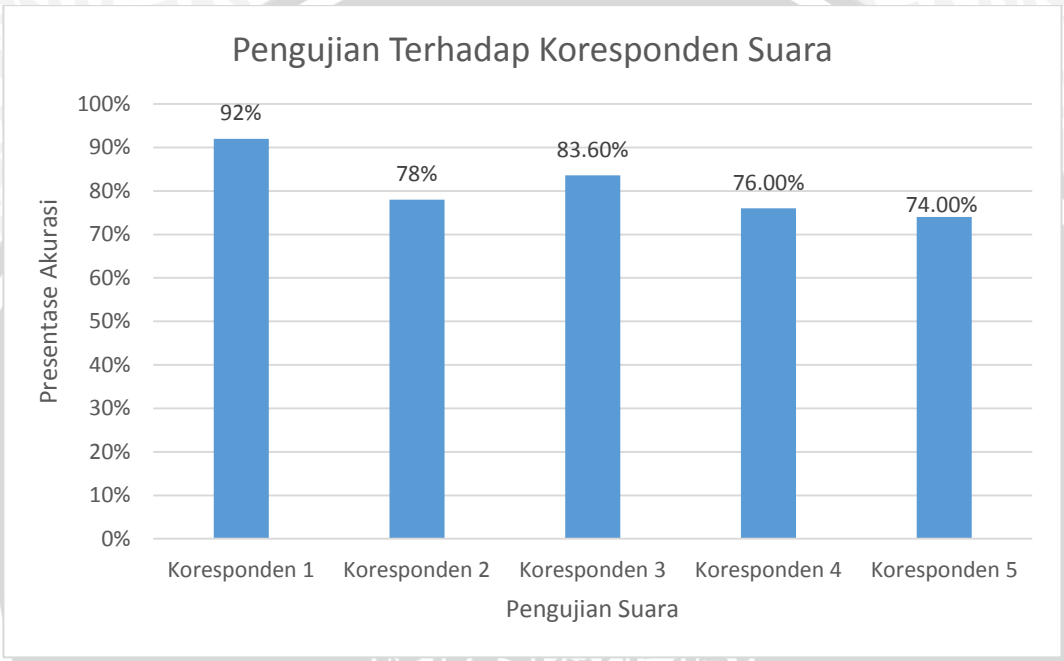
12	Kunci	410	410	410	410
13	Lampu	410	410	410	410
14	Lihat	410	410	410	410
15	Main	610	410	410	476.67
16	Matikan	510	410	410	443.33
17	Pagar	510	410	410	443.33
18	Pintu	410	510	510	476.67
19	Rumah	510	510	510	510
20	Sakit	610	610	410	543.33
21	Satu	610	610	610	610
22	Segelas	410	310	510	410
23	Telepon	610	610	610	610
24	Tolong	580	610	610	600
25	Tutup	610	600	580	596.67

Keterangan : - ms (milidetik)

- Setiap pengucapan kata pada pengujian waktu dimulai pada detik ke 0

5.3. Hasil Pengujian Validasi

Analisis pengujian validasi berfungsi untuk dilihat kesesuaian antara hasil implementasi dengan kebutuhan pengguna. Dari pengujian diatas, dapat disimpulkan bahwa hasil implementasi dari segi validasi telah memenuhi kebutuhan pengguna karena dari masing-masing koresponden suara telah menghasilkan akurasi diatas 74%.

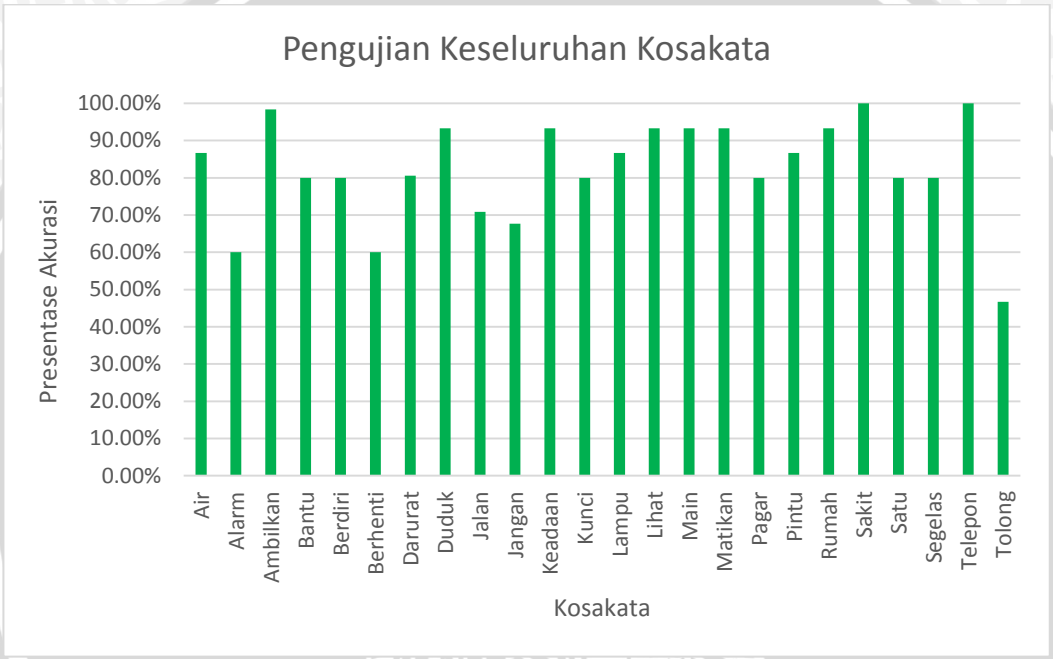


Gambar 5.3 Grafik Pengujian Koresponden Suara

Berdasarkan gambar 5.3, Koresponden pertama berhasil mengucapkan 69 kosakata dari 25 jumlah kosakata yang telah diucapkan sebanyak 3 kali sehingga koresponden pertama melakukan 75 pengucapan dan memiliki akurasi sebesar 92%. Koresponden kedua berhasil mengucapkan 58 kosakata dari 25 jumlah kosakata yang telah diucapkan sebanyak 3 kali sehingga koresponden kedua melakukan 75 pengucapan dan memiliki akurasi sebesar 78%. Koresponden ketiga berhasil mengucapkan 63 kosakata dari 25 jumlah kosakata yang telah diucapkan sebanyak 3 kali sehingga koresponden ketiga melakukan 75 pengucapan dan memiliki akurasi sebesar 83.60%. Koresponden keempat berhasil mengucapkan 57 kosakata dari 25 jumlah kosakata yang telah diucapkan sebanyak 3 kali sehingga koresponden ketiga melakukan 75 pengucapan dan

memiliki akurasi sebesar 76%. Koresponden kelima berhasil mengucapkan 55 kosakata dari 25 jumlah kosakata yang telah diucapkan sebanyak 3 kali sehingga koresponden kelima melakukan 75 pengucapan dan memiliki akurasi sebesar 74%.

Selanjutnya akan dijabarkan hasil akurasi setiap kata berdasarkan semua hasil percobaan aplikasi yang dilakukan oleh koresponden suara. Hasil pengujian tersebut dijabarkan oleh gambar 5.4.



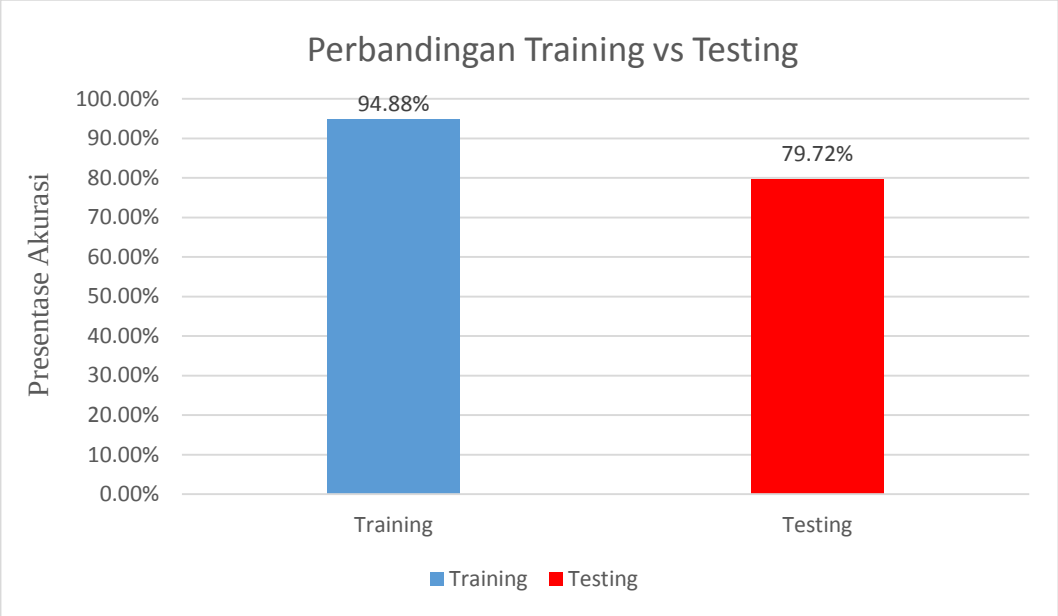
Gambar 5.4 Akurasi Keseluruhan Kosakata

5.4. Analisis & Evaluasi Hasil Program

Hasil training suara dapat berbeda-beda walaupun jumlah kalimat dan file suara tetap sama karena perubahan parameter dan nilai-nilai variabel sangat mempengaruhi hasil training yang akan berdampak pula pada hasil uji coba pengenalan suara. Penulis mengambil hasil *training* & *testing* terbaik lalu dituangkan dalam laporan.

Dari hasil pengujian di atas, dihitung rata-rata berdasarkan jumlah nilai benar pada saat tahap pengenalan suara yang ditampilkan dalam Gambar 5.4. Dari tingkat akurasi diatas, dapat dinyatakan bahwa keterbatasan produk PocketSphinx pada pembuatan *acoustic model* secara manual memberikan hasil akurasi sekitar

94.88% berdasarkan 241 kosakata yang dikenali dari total 254 kosakata yang tersedia untuk keseluruhan pengujian pemilihan parameter terbaik. Hasil *training* didapat dari hasil akurasi perekaman yang didapat berdasarkan pemilihan parameter terbaik **Lampiran 2**. Sedangkan, pada hasil *testing* didapat dari jumlah akurasi koresponden sebesar 79,72% dengan perincian, koresponden pertama memiliki akurasi sebesar 92%, koresponden kedua memiliki akurasi sebesar 78%, koresponden ketiga memiliki akurasi sebesar 84%, koresponden keempat memiliki akurasi sebesar 76%, koresponden kelima memiliki akurasi sebesar 74% yang mencoba langsung aplikasi.



Gambar 5.5 Grafik Perbandingan Akurasi *Training* dan *Testing*

Peneliti mencoba untuk memodifikasi parameter yang dapat diubah sehingga dapat menentukan hasil yang terbaik untuk program dalam menangkap sinyal suara dan pola suara orang Indonesia, berikut ini hasilnya berdasarkan perbandingan parameter-parameternya.

Setelah melihat hasil training dan hasil uji coba berdasarkan beberapa faktor yang telah diuji, dapat disimpulkan bahwa terdapat beberapa hal yang dapat mempengaruhi program dalam menangkap sinyal suara. Hal ini terjadi karena beberapa faktor yaitu:

- Perekaman suara tidak semua dilakukan di ruang rekaman khusus.
- Kosakata agak sulit diucapkan karena harus mengikuti pola ARPAbet Inggris - Amerika.
- Terdapat beberapa kosakata yang pengucapannya hampir mirip berdasarkan fonemnya sehingga sulit dikenali dengan baik oleh aplikasi.

Disamping itu masih terdapat kelemahan lainnya pada aplikasi pengenalan *voice command* berbahasa Indonesia yaitu keakuratan teks yang ditampilkan dapat menurun jika suara yang masuk bercampur dengan banyak gangguan suara. Perbedaan intonasi dan cara pelafalan yang kurang mendekati suara pada saat training akan membuat aplikasi menampilkan teks yang salah. Hal ini menyebabkan pembicara terkadang harus mengulang pengucapannya untuk mendapatkan kosakata yang sesuai.

Dengan hasil training pengenalan suara versi bahasa Indonesia yang mencapai 94.88% oleh 7 koresponden suara berbeda dan untuk hasil testing program yang dilakukan oleh 5 koresponden berbeda total memiliki akurasi mencapai 79%. Penulis yakin aplikasi *voice command speech to text* ini dapat terus dikembangkan lebih baik lagi dengan menambah jumlah kosakata, variasi suara dan mengestimasi parameter-parameter dalam proses training suara yang sangat mempengaruhi hasil pengenalan suara.

BAB VI
PENUTUP

Pada bagian ini akan dipaparkan kesimpulan dan saran yang dapat diambil setelah melakukan pengujian terhadap sistem.

6.1. Kesimpulan

Kesimpulan yang dapat diambil setelah melakukan pengujian terhadap aplikasi *voice command speech to text* antara lain:

1. Cara merancang aplikasi dengan fitur pengenalan *voice command* Berbahasa Indonesia berbasis *offline* untuk mengenali pola suara orang Indonesia dengan menggunakan *Library* PocketSphinx dan penggunaan metode *Hidden Markov Model.*
2. Cara mengatur *library* PocketSphinx agar dapat mengenali suara Bahasa Indonesia dengan mengubah kamus pengucapan yang disesuaikan dengan fonem pengucapan orang Indonesia dan mengestimasi parameter-parameter pada saat melakukan *training* suara.
3. Karakter suara orang Indonesia dapat diidentifikasi dengan cara melakukan training suara sehingga menghasilkan *acoustic model* secara manual yang memberikan akurasi sebesar 94.88% berdasarkan 20 basis data suara dari 7 suara orang Indonesia yang berbeda berdasarkan parameter terbaik yang mampu menekan word error rate (WER) pada saat *training* suara dengan mengubah nilai sebagai berikut:
    - *NFFT = 512*
    - *NFIL = 20*
    - *Lowerf = 1Hz*
    - *Upperf = 4000Hz*
4. Perfomansi pada saat implementasi pengenalan suara bahasa Indonesia memiliki tingkat akurasi 79% berdasarkan suara dari 5 orang yang berbeda.

6.2. Saran

Dari penulis bagi para peneliti atau programmer lain untuk memperbaiki dan meningkatkan kualitas aplikasi pengenalan *voice command* berbahasa Indonesia.

1. Pengimplementasian aplikasi masih menggunakan *library* dengan database bahasa Inggris sehingga kata-kata yang terdapat di kamus pengucapan harus disesuaikan dengan lafal orang Indonesia. Diharapkan untuk penelitian selanjutnya dapat membuat library yang dapat mengenali pola pengucapan bahasa Indonesia murni.
2. Penambahan perbendaharaan kata, perekaman suara harus dilakukan di ruangan yang kedap suara agar tidak banyak *noise* yang ikut terekam dan harus menggunakan *microphone* yang kualitasnya bagus supaya suara terdengar lebih jelas.

DAFTAR PUSTAKA

[MSD-12] Marietha, Sonya, Ayu Purwarianti, dan Dessi Puji Lestari. 2012, "SMSsuara Application with Automatic Speech Recognition and Text to Speech on Mobile Phone", Jurnal Sarjana ITB bidang Teknik Elektro dan Informatika Vol.1, No.1, hal 39-43.

[ANO-14] Anonymous.http://www.speech.cs.cmu.edu/comp.speech/Section6/Q6.1.htm. Diakses Pada tanggal 16 Juli 2014.

[AMT-11] Ahsan, K. M. T. 2011, "Implementation of Bangla Speech Recognition System on Cell Phones", School of Engineering and Computer Science, BRAC University

[ARI-12] Amira, Hapsari M. 2012, "Sistem Layanan Reservasi Menggunakan Ucapan Bahasa Indonesia", Jurnal Sarjana ITB bidang Teknik Elektro dan Informatika, Vol. 1, No.1, hal 33-38.

[LRA-89] Lawnrence R. Rabiner, "A Tutorial on Hiden Markov Model and selected Aplications in specch recognition", 77:2, (February 1989), 257-259.

[HRV-14] Handel, R., V, 2008, "Hidden Markov Models". Diperoleh 23 Juli 2014 dari https://www.princeton.edu/~rvan/orf557/hmm080728.pdf

[MAM-10] Mohammad A. M. Abushariah, dkk. 2010,"Natural Speaker-Independent Arabic Speech Recognition System Based on Hidden Markov Models Using Sphinx Tools". International Conference on Computer and Communication Engineering (ICCCE 2010).

[FVA-11] Ferdiansyah, Veri, dan Ayu Purwarianti. 2011, "Indonesian Automatic Speech Recognition System using English-based Acoustic Model", American Journal of Signal Processing 2012, Vol. 2, No.4, hal 60-63.

[MMI-10] Muliati, Mira. 2010, "Pemeriksaan Urutan Hafalan Al Quran Memanfaatkan Pengenalan Suara Otomatis", KNIF 2010.

[SHH-09] Satori, H., Hiyassat, H., Harti, M., & Chenfour, N. 2009, "Investigation Arabic Speech Recognition Using CMU Sphinx System". Diperoleh 23 Juli 2014 dari

http://www.ccis2k.org/iajit/PDF/vol.6,no.2/11IASRUCSS186.pdf.

[CSA-10] Chowdhury, Shammur A. 2010, "Implementation of Speech Recognition System for Bangla", Department of Computer Science and Engineering, BRAC University.

[CMU-14] Anonymous. Carnegie Mellon University. Diperoleh 23 Juli 2014 dari http://cmusphinx.sourceforge.net/wiki/tutorial*PocketSphinx*/.

[SPH-14] Anonymous.Carnegie Mellon University. Diperoleh 23 Juli 2014 dari (http://www.speech.cs.cmu.edu/sphinx/tutorial.html).

[ACM-14] Anonymous.Carnegie Mellon University. Diperoleh 23 Juli 2014 dari (http://cmusphinx.sourceforge.net/wiki/tutorialam).

[ACG-14] Anonymous.Carnegie Mellon University. Diperoleh 23 Juli 2014 dari (http://www.speech.cs.cmu.edu/cgi-bin/cmudict).

[AMN-14] Anonymous.Carnegie Mellon University. Diperoleh 23 Juli 2014 dari (http://www.speech.cs.cmu.edu/sphinxman/scriptman1.html)

[KGB-13] I Kadek Suryadharma, Gelar Budiman, ST., MT. Budhi Irawan, Ssi.,MT., 2013, "Perancangan Aplikasi Speech To Text Bahasa Inggris Ke Bahasa bali Menggunakan *PocketSphinx* Berbasis Android".

[JAM-05] Jackson, M. 2005, "Automatic Speech Recognition: Human Computer Interface For Kinyawanda Language, Computer Science of Makerere University.

[FLC-14] Feng L, C. "Understanding the CMU Sphinx Speech Recognition System", Department of Computer Science, National Chengchi University.

[TSA-13] Wijaya, Tony, Samuel Susanto. 2013, "Speech Recognition Bahasa Indonesia untuk Android", Universitas Bina Nusantara, Jakarta.

[MVE-12] Monika, V. 2012, "Perancangan Program Aplikasi Android Speech To Text Bahasa Indonesia dan Inggris Menggunakan Metode Hidden Markov Model", Universitas Binus.

[MUB-10] Munawar, B. 2010-08-20, "Word Identification Using Hidden Markov Model (HMM) Through Feature Extraction Linear Predictive Coding (LPC)", Undergraduate Theses from

Perpustakaan UNIKOM.

[EBP-10] Eko Budi Prasetyo, M. 2010. "Teori Dasar Hidden Markov Model", Program Studi Sistem dan Teknologi Informasi, Institut Teknologi Banding.

[NKA-10] Noora M. Rodliyah, I Ketut Eddy Purnama, dan Ahmad Zaini. 2010. "Speech To Video (S2v), Sistem Komunikasi Portabel Untuk Penderita Tunarungu", Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember.

LAMPIRAN

Lampiran 1

The ARPAbet

Vowels		
Phoneme	**Example**	**Transcription**
AA	bot	B AA T
AE	bat	B AE T
AH	but	B AH T
AO	bought	B AO T
AW	bout	B AW T
AY	bite	B AY T
EH	bet	B EH T
ER	bird	B ER D
EY	bait	B EY T
IH	bit	B IH T
IY	beat	B IY T
OW	boat	B OW T
OY	boy	B OY
UH	put	P UH T
UW	boot	B UW T

Consonants		
Phoneme	**Example**	**Transcription**
B	be	B IY
CH	cheese	CH IY Z
D	day	D EY
DH	that	TH AE T
F	fee	F IY
G	go	G OW
HH	he	HH IY
JH	just	JH AH S T
K	key	K IY
L	late	L EY T
M	me	M IY
N	knee	N IY
NG	sing	S IH NG
P	pay	P EY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	thanks	TH AE NG K S
V	vain	V EY N
W	we	W IY
Y	yes	Y EH S
Z	zoo	Z UW
ZH	pleasure	P L EH ZH ER

Stress ini mempunyai beberapa tingkatan level dalam pengucapan di ARPAbet

Value	Level of stress
0	no stress
1	primary stress
2	secondary stress

Contoh:

in	AH0 N, IH1 N
the	DH AH0, DH AH1, DH IY0
dictionary	D IH1 K SH AH0 N EH2 R IY0
stress	S T R EH1 S
is	IH1 Z, AH0 Z
indicated	IH1 N D AH0 K EY2 T AH0 D
by	B AY1
digits	D IH1 JH AH0 T S
following	F AA1 L OW0 IH0 NG
stressed	S T R EH1 S T
vowels	V AW1 AH0 L Z

Sumber: The CMU Pronouncing Dictionary - Speech at CMU (http://www.speech.cs.cmu.edu/cgi-bin/cmudict)

Lampiran 2

matikan nyalakan lampu alarm bukakan tutup pintu ambilkan rumah telepon (SPEAKER_1-A1)
matikan nyalakan lampu alarm bukakan tutup pintu ambilkan rumah telepon (SPEAKER_1-A1)
Words: 10 Correct: 10 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
buat ambilkan segelas air lihat keadaan tolong bantu lempar saya panggilan darurat (SPEAKER_2-A2)
buat ambilkan segelas air lihat keadaan tolong bantu lempar saya panggilan darurat (SPEAKER_2-A2)
Words: 12 Correct: 12 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
selamat pagi malam siang sore sekarang hujan kering mendung kunci pagar (SPEAKER_3-A3)
selamat pagi malam siang sore sekarang hujan kering mendung kunci pagar (SPEAKER_3-A3)
Words: 11 Correct: 11 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
satu dua tiga empat lima enam tujuh delapan sembilan sepuluh mesin (SPEAKER_4-A4)
satu dua tiga empat lima enam tujuh delapan sembilan sepuluh mesin (SPEAKER_4-A4)
Words: 11 Correct: 11 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
kemana bagaimana bila apa dimana siapa mengapa kapan darimana berapa atau (SPEAKER_5-A5)
kemana bagaimana bila apa dimana siapa mengapa kapan darimana berapa atau (SPEAKER_5-A5)
Words: 11 Correct: 11 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
jalan berhenti sakit sembuh mimpi main tidur pergi jangan duduk berdiri (SPEAKER_6-A6)
jalan berhenti sakit sembuh mimpi main tidur pergi jangan duduk berdiri (SPEAKER_6-A6)
Words: 11 Correct: 11 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
keadaan bantu selamat sekarang mendung mengapa kapan darimana berhenti sembuh tidur berdiri (SPEAKER_7-A7)
keadaan bantu selamat sekarang mendung mengapa kapan darimana berhenti sembuh tidur berdiri (SPEAKER_7-A7)
Words: 12 Correct: 12 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
mimpi sakit KEMANA siapa satu sembilan mesin lempar ambilkan tutup matikan lampu pagar (SPEAKER_8-A8)
mimpi sakit DIMANA siapa satu sembilan mesin lempar ambilkan tutup matikan lampu pagar (SPEAKER_8-A8)
Words: 13 Correct: 12 Errors: 1 Percent correct = 92.31% Error = 7.69% Accuracy = 92.31%
Insertions: 0 Deletions: 0 Substitutions: 1
mesin pagar berhenti pintu rumah segelas telepon tolong darurat duduk jalan jangan (SPEAKER_9-A9)
mesin pagar berhenti pintu rumah segelas telepon tolong darurat duduk jalan jangan (SPEAKER_9-A9)
Words: 12 Correct: 12 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
SEPULUH sembilan DELAPAN tujuh enam lima empat tiga dua satu alarm bukakan kering (SPEAKER_10-A10)

SATU sembilan TELEPON tujuh enam lima empat tiga dua satu alarm bukakan kering (SPEAKER_10-A10)
Words: 13 Correct: 11 Errors: 2 Percent correct = 84.62% Error = 15.38% Accuracy = 84.62%

Insertions: 0 Deletions: 0 Substitutions: 2
main kunci keadaan jalan berdiri duduk lempar berdiri lempar bantu lihat ambilkan tolong (SPEAKER_11-A11)
main kunci keadaan jalan berdiri duduk lempar berdiri lempar bantu lihat ambilkan tolong (SPEAKER_11-A11)
Words: 13 Correct: 13 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
pagar mesin matikan pergi pintu rumah sakit telepon lampu berdiri duduk segelas air kunci (SPEAKER_12-A12)
pagar mesin matikan pergi pintu rumah sakit telepon lampu berdiri duduk segelas air kunci (SPEAKER_12-A12)
Words: 14 Correct: 14 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
lampu tolong segelas telepon sakit rumah pintu pergi PAGAR mesin matikan main keadaan (SPEAKER_13-A13)
lampu tolong segelas telepon sakit rumah pintu pergi KEADAAN mesin matikan main keadaan (SPEAKER_13-A13)
Words: 13 Correct: 12 Errors: 1 Percent correct = 92.31% Error = 7.69% Accuracy = 92.31%
Insertions: 0 Deletions: 0 Substitutions: 1
kering kunci lampu lempar bantu berhenti berdiri darurat tutup matikan sembuh tidur segelas (SPEAKER_14-A14)
kering kunci lampu lempar bantu berhenti berdiri darurat tutup matikan sembuh tidur segelas (SPEAKER_14-A14)
Words: 13 Correct: 13 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
mesin tutup bukakan matikan nyalakan lampu alarm LEMPAR BANTU LIHAT rumah pintu pergi pagar (SPEAKER_15-A15)
mesin tutup bukakan matikan nyalakan lampu alarm *** APA TIGA rumah pintu pergi pagar (SPEAKER_15-A15)
Words: 14 Correct: 11 Errors: 3 Percent correct = 78.57% Error = 21.43% Accuracy = 78.57%
Insertions: 0 Deletions: 1 Substitutions: 2
mesin pagar berhenti pintu rumah lampu berdiri duduk segelas air kunci rumah pintu pergi pagar (SPEAKER_16-A16)
mesin pagar berhenti pintu rumah lampu berdiri duduk segelas air kunci rumah pintu pergi pagar (SPEAKER_16-A16)
Words: 15 Correct: 15 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
selamat pagi malam siang sore sekarang hujan kering mendung berhenti sembuh tidur MAIN KERING (SPEAKER_17-A17)
selamat pagi malam siang sore sekarang hujan kering mendung berhenti sembuh tidur *** MIMPI (SPEAKER_17-A17)
Words: 14 Correct: 12 Errors: 2 Percent correct = 85.71% Error = 14.29% Accuracy = 85.71%
Insertions: 0 Deletions: 1 Substitutions: 1
PINTU PERGI pagar mesin matikan main keadaan mesin lempar ambilkan tutup matikan rumah telepon (SPEAKER_18-A18)
*** MATIKAN pagar mesin matikan main keadaan mesin lempar ambilkan tutup matikan rumah telepon (SPEAKER_18-A18)
Words: 14 Correct: 12 Errors: 2 Percent correct = 85.71% Error = 14.29% Accuracy = 85.71%
Insertions: 0 Deletions: 1 Substitutions: 1

PINTU rumah segelas TELEPON tolong lihat keadaan tolong bantu lempar saya berhenti sembuh tidur (SPEAKER_19-A19)
LAMPU rumah segelas AIR tolong lihat keadaan tolong bantu lempar saya berhenti sembuh tidur (SPEAKER_19-A19)
Words: 14 Correct: 12 Errors: 2 Percent correct = 85.71% Error = 14.29% Accuracy = 85.71%
Insertions: 0 Deletions: 0 Substitutions: 2
segelas sekarang selamat sembilan sembuh sepuluh siang siapa sore telepon tidur tiga matikan mesin (SPEAKER_20-A20)
segelas sekarang selamat sembilan sembuh sepuluh siang siapa sore telepon tidur tiga matikan mesin (SPEAKER_20-A20)
Words: 14 Correct: 14 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
TOTAL Words: 254 Correct: 241 Errors: 13
TOTAL Percent correct = 94.88% Error = 5.12% Accuracy = 94.88%
TOTAL Insertions: 0 Deletions: 3 Substitutions: 10

Sumber: Pelatihan Suara menggunakan *SphinxTrain*