RANCANG BANGUN APLIKASI PENGENALAN UCAPAN UNTUK KATA DALAM HADITS BERBAHASA ARAB DENGAN *FRAMEWORK SPHINX*

SKRIPSI

Oleh:
NAJA IKMAL NAJIB
NIM. 11650047

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

RANCANG BANGUN APLIKASI PENGENALAN UCAPAN UNTUK KATA DALAM HADITS BERBAHASA ARAB DENGAN *FRAMEWORK SPHINX*

SKRIPSI

Diajukan Kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri
Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S. Kom.)

Oleh:
NAJA IKMAL NAJIB
NIM. 11650047

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

RANCANG BANGUN APLIKASI PENGENALAN UCAPAN UNTUK KATA DALAM HADITS BERBAHASA ARAB DENGAN *FRAMEWORK SPHINX*

SKRIPSI

Oleh:
NAJA IKMAL NAJIB
NIM. 11650047

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: .....…

Pembimbing I	Pembimbing II
Totok Chamidy, M. Kom.	**Dr. Suhartono, M. Kom.**
NIP. 19691222 200604 1 001	**NIP. 19680519 200312 1 001**

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian, M. CS.
NIP. 19740424 200901 1 008

RANCANG BANGUN APLIKASI PENGENALAN UCAPAN UNTUK KATA DALAM HADITS BERBAHASA ARAB DENGAN *FRAMEWORK SPHINX*

SKRIPSI

Oleh:
NAJA IKMAL NAJIB
NIM. 11650047

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S. Kom.)
Tanggal: .....…

Susunan Dewan Penguji:		**Tanda Tangan**
1. Penguji Utama:	**Dr. Muhammad Faisal, M. T.** **NIP. 19740510 200501 1 007**	**( )**
2. Ketua:	**Fatchurrochman, M. Kom.** **NIP. 19700731 200501 1 002**	**( )**
3. Sekretaris:	**Totok Chamidy, M. Kom.** **NIP. 19691222 200604 1 001**	**( )**
4. Anggota:	**Dr. Suhartono, M. Kom.** **NIP. 19680519 200312 1 001**	**( )**

Mengetahui dan Mengesahkan
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian, M. CS.
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama:	Naja Ikmal Najib
NIM:	11650047
Jurusan:	Teknik Informatika
Fakultas:	Sains dan Teknologi

menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis ini benar-banar merupakan hasil karya saya sendiri, bukan merupakan pengambilalihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, tanpa mencantumkan sumber cuplikan pada daftar pustaka.

Apabila di kemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia mempertanggungjawabkan perbuatan saya sesuai peraturan yang berlaku.

Malang, 16 Juni 2016

Yang Membuat Pernyataan,

NAJA IKMAL NAJIB

NIM. 11650047

KATA PENGANTAR

Alhamdulillah, segala puji hanya bagi Allah, Tuhan Yang Maha Esa, atas segala limpahan rahmatnya. Sholawat juga dilimpahkan kepada Nabi Muhammd Alaihi as-sholat wa as-salam sebagai bentuk syukur penulis kepada Allah, yang dapat membantu menyelesaikan permasalahan tugas akhir berupa skripsi ini.

Penulis juga mengucapkan terima kasih kepada kedua orang tua yang terus memberikan semangat kepada anak-anaknya agar penulisan laporan tugas akhir ini dapat segera direalisasikan. Ucapan terima kasih juga penulis sampaikan kepada kedua dosen pembimbing yang memotivasi penulis agar dapat menyelesaikan tugas akhir ini sesuai dengan waktu yang diberikan.

Ide tentang skripsi ini sejatinya sederhana. Penulis mendapatkannya setelah terinspirasi oleh kinerja Google Translate yang memiliki fitur pengenalan ucapan (speech recognition). Fleksibilitas kinerja peranti penerjemah yang dikembangkan Google itu membuat penulis terdorong untuk mengimplementasikannya ke dalam bentuk source code Java. Awalnya penulis merasa kesulitan dalam mencari tools yang tepat untuk membantu merancang bangun sistem yang diinginkan penulis. Kemudian, setelah salah seorang dosen pembimbing menyarankan sebuah tools yang dimaksud, penulis melakukan eksplorasi lebih jauh tentang cara kerja tools yang bersangkutan. Hasilnya juga cukup memuaskan, walau hanya sekadar contoh program. Sayangnya, di awal percobaan yang dilakukan penulis, tools yang diberikan hanya mencantumkan data penunjang yang berasal dari Bahasa Inggris, bukan seperti yang dikehendaki. Akhirnya, penulis mulai melakukan desain sistem tersebut dari nol.

Kasus yang diangkat dalam laporan ini adalah Hadits. Sebuah kasus yang jarang diimplementasikan dalam sistem pengenalan ucapan. Sejauh pengamatan penulis, kasus yang dimuat dalam jurnal pengenalan ucapan berbahasa Arab lebih dominan kepada pembacaan angka dan al-Qur'an, baik dari segi lafal maupun sekadar perintah pembacaan. Itulah yang menjadi motivasi penulis untuk mengangkat kasus ini di bidang pengenalan ucapan.

Penulis berharap penelitian ini tidak hanya menjadi refleksi tentang kinerja pengenalan ucapan secara umum, melainkan kontribusi terkait pengembangan sistem yang lebih lanjut, sehingga semakin banyak kaum muslimin yang sadar akan pentingnya kitab-kitab hadits setelah al-Qur'an. Di samping itu, secara umum penulis juga berharap kaum muslimin segera mengejar ketertinggalan dari berbagai aspek kehidupan, termasuk di bidang sains dan teknologi. Sehingga tidak mudah dimanfaatkan oleh kaum lain yang memusuhi Islam karena anggapan bahwa Islam identik oleh musuh-musuhnya dengan sifat kolot dan kuno yang dialami secara bertahun-tahun.

Apa yang tersaji di dalam penelitian ini merupakan hasil kerja keras penulis yang diupayakan selama berbulan-bulan. Memang tidak ada penulisan yang bebas kekurangan dalam laporan ini. Namun, setidaknya mendorong kaum muslimin yang peduli dengan perkembangan sains dan teknologi, terutama teknik informatika ini sehingga dapat menelaah kembali, membuktikan dengan pengujian, dan diakhiri dengan kesimpulan yang konstruktif. Dengan demikian, perkembangan ilmu pengetahuan di bidang sains (terutama pengolahan ucapan) dapat berlangsung dengan cepat dan kaum muslimin (secara umum) dapat menanggapi dinamika perkembangan berbagai disiplin ilmu dengan cepat dan tepat pula.

Akhir kata, penulis menyampaikan terima kasih sekali lagi kepada berbagai pihak yang telah membantu menyelesaikan tugas akhir ini. Kemudian, penulis menyampaikan permohonan maaf jika terjadi kekeliruan dalam penulisan laporan skripsi ini. Kritik dan saran sangat dibutuhkan agar implementasi sistem ini dapat berjalan sebagaimana mestinya, sehingga dapat dimanfaatkan dengan baik dan benar oleh kaum muslimin secara luas.

Wassalamualaikum wr. wb.

Malang, 16 Juni 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Najib, Naja Ikmal. 2016. **RANCANG BANGUN APLIKASI PENGENALAN UCAPAN UNTUK KATA DALAM HADITS BERBAHASA ARAB DENGAN *FRAMEWORK SPHINX*** . Skripsi. Jurusan Teknik Informatika. Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: **(i) Totok Chamidy, M. Kom, (ii) Dr. Suhartono, M. Kom**

Kata Kunci: Hidden Markov Model, Sphinx-4, Pengenalan Ucapan, Hadits, Language Model, Acoustic Model

Abstrak

Konsep sistem pengenalan ucapan telah banyak dikembangkan selama puluhan tahun sejak berbagai bentuk algoritma dirumuskan untuk membantu mempermudah dan mempercepat proses komputasi. Salah satu metode yang terkenal dalam sistem pengenalan ucapan adalah Hidden Markov Model (HMM). Meski demikian, berbagai mayoritas literatur yang terbit sejak dekade 70-80 masih membahas penjabaran metode yang berkutat di dalam konsep perhitungan yang rumit, sehingga penjabaran konsep di dalam proses komputasi masih sangat terbatas di kalangan ilmuwan komputer. Di samping itu, ilmuwan komputer di masa itu menemui keterbatasan dalam menyediakan sumber daya yang cukup untuk mengadakan pelatihan tentang perancangan sistem pengenalan ucapan. Hal-hal tersebut menjadi sebab utama penerapan sistem pengenalan ucapan dianggap hal baru pada masa kini.

Dalam laporan ini, penulis menawarkan sebuah framework yang dipakai dalam membantu merancang sistem pengenalan ucapan yang berbasis java, yaitu Sphinx- 4. Sphinx-4 telah dirancang oleh ilmuwan komputer dari Universitas Carnegie- Mellon (CMU) sejak awal tahun 90-an, bekerjasama dengan Mitsubishi Electric Research Laboratory, dan Sun Microsystem. Model yang dibutuhkan agar sistem ini dapat berjalan adalah Language Model, Dictionary, dan Acoustic Model Model inilah yang akan menjadi acuan yang memudahkan penulis dalam merancang bangun aplikasi pengenalan ucapan berbasis Java.

Dalam Islam, sumber pedoman yang paling banyak dirujuk setelah al-Qur'an adalah Hadits. Namun, dalam praktik rancang bangun sistem pengenalan ucapan, kasus yang lebih dominan ada pada pengucapan angka Arab dan al-Qur'an. Sedangkan untuk hadits masih jarang diangkat di kalangan ilmu pengenalan ucapan. Oleh karena itu, kasus yang akan diangkat dalam penelitian ini adalah pengenalan ucapan dalam Hadits Berbahasa Arab

Najib, Naja Ikmal. 2016. **DESIGNING SPEECH RECOGNITION APPLICATION FOR THE WORDS IN ARABIC HADITS WITH SPHINX FRAMEWORK**. Thesis. Informatic Engineering Faculty of Science and Technology Islamic State University of Maulana Malik Ibrahim Malang. Advisor: (I) Totok Chamidy, M. Kom, (II) Dr. Suhartono, M. Kom

Keywords: Hidden Markov Model, Sphinx-4, Speech Recognition, Hadits, Language Model, Acoustic Model

Abstract

Speech Recognition system had been developed many years since many kinds of algorithms were formulated to help, to ease, and to speed up computation processes. One of the most renowed method in speech recognition system is Hidden Markov Model (HMM). However, majority of published literature since 70-80 th decade had still reviewed the method explanation that were still holding on complicated calculation concept; as a result, their concepts analysis in computation processes are stil restricted in computer scientist. Besides, computer scientist in that past time had faced many limitation in providing adequate resources to establish a training about designing speech recognition system. Those mentioned matters become main cause why speech recognition system implementation is considered as a something new in this recent days.

In this paper, writer has offered a framework that will be used to help designing java-based speech recognition system: Sphinx-4. Sphinx-4 had been developed by computer scientist from Carnegie Mellon University (CMU) since early 90's, in collaboration with Mitsubishi Electric Research Laboratory, and Sun Microsystem. Model required to make this system work is Language Model, Dictionary, and Acoustic Model. These models will be reference which facilitate writer in designing Java-based speech recognition application.

In Islam, most referred guidance after Quran is Hadits. But, in practice of speech recognition, the most dominant case that emerge in research is Arabic Numerical Dictation and Qur'an. Whereas, hadits is rarely reviewed in Digital Speech Processing. Therefore, the case that will be raised in this research is Speech Recognition in Arabic Hadits

النجا نجيب تصميم تطبيقات التعرف على الكلام عن الكلمات في اللغة العربية التحدث مع إكمال. 2016 .
مولنا الحكمية السإلمية الجامعة والتكنولوجيا للعلوم الهندسإة كلية المعلوماتية.SPHINX الإطار الحديث
إبراهيم مالنج. توتةك)واحد(مستشار: حميدي، م. الدكتور)الثاني(كوم، سإوهرتونو، كوم م. أطروحة.
مالك

الـ : المخفية ماركوف أبههو النموذجي، الههول- 4 ، علهى التعههرف الكلم، اللغهة حههديث، النمههوذجي،
بحث كلمات الصوتية نموذج

تجريدية فكرة

خوارزميات أنواع من العديد صيغت منذ عديدة سإنوات الكلم على التعرف نظام تطوير تم وقد للمساعدة، عمليات وتسريع لتخفيف، هو الكلم على التعرف نظام في المشهورة الطرق أكثر من واحدة حساب. المخفية ماركوف ومع النموذجي. عقود منذ المنشورة الكتابات معظم ذلك، 70 - 80 اسإتعرض يزال ل كان حسابية مفهوم عقد على تزال ل كانت التي الطريقة شرح ونتيجة معقدة. في مفاهيمها تحليل تقييد يتم لذلك، عالم في خارج يزال الحسابية العمليات جانب الى الكمبيوتر. الوقت ذلك في الكمبيوتر عالم كان ذلك، التعرف نظام تصميم حول تدريبية دورة لقامة الكافية الموارد توفير في قيود من العديد واجهت الماضي على الخيرة الكلم على التعرف نظام تنفيذ يعتبر لماذا الرئيسي السبب المذكورة المسائل هذه تصبح الكلم. باعتباره اليام هذه في جديدا شيئا.

هذه في الكلم على التعرف نظام تصميم في للمساعدة اسإتخدامها سإيتم التي إطار الكاتب وعرضت الورقة، أبو القائمة: الهول- 4 . أبو وكان الهول- 4 ميلون كارنيجي جامعة من الكمبيوتر عالم قبل من وضعها تم مبكر وقت منذ ميلون(كارنيجي)جامعة 90 ، الكهربائية أبحاث مختبر مع بالتعاون جافا ميتسوبيشي، اللغة نظام هو العمل هذا لجعل المطلوبة نموذج والميكروسإيستم. النموذجي، والصوتية قاموس، نموذج. على التعرف تطبيق على القائمة جافا تصميم في الكاتب تسهل التي المرجعية النماذج هذه تكون سإوف .الكلم على التعرف من العملية الناحية من ولكن، حديث. هو القرآن بعد التوجيه المشار ومعظم السإلم، في الملءا العددية العربية هي البحث في تنشأ التي البرز والقضية الكلم، في والقرآن. ما نادرا حديث حين، الكلم معالجة في اسإتعرضت الرقمية. على التعرف هو البحث هذا في سإتطرح التي القضية فإن ولذلك، .الكلم العربية حديث في

BAB I

PENDAHULUAN

1.1. Latar Belakang

Sebagai makhluk yang beragama, manusia senantiasa akan merujuk kepada sumber ajaran yang utama sebagai fondasi yang mengokohkan pendiriannya ketika dihadapkan pada berbagai permasalahan hidup. Tidak hanya pada saat menempuh pendidikan formal, ketika memasuki dunia kerja, tidak sedikit manusia yang menyadari akan pentingnya menjaga sumber-sumber ajaran yang mengarahkannya menuju makhluk yang didambakan oleh Tuhannya, mengingat kehidupan yang masih dan akan diarungi tidak berkutat pada hal-hal yang terbatas pada masalah terkini dan 'yang hanya ada di sini'. Kesadaran inilah yang menyebabkan manusia berupaya mengenal siapa jati dirinya, terlepas dari apa yang diinginkannya, mulai dari awal kelahirannya hingga masa depannya nanti setelah meninggal dunia. Inilah yang ditekankan oleh ajaran Islam.

Agama ini telah diwasiatkan oleh Allah SWT melalui Rasul Allah Muhammad untuk dipegang teguh, dijaga, dan dipertahankan oleh pengikutnya hingga ajal menjemput. Untuk menjaganya, perlu bimbingan Allah SWT melalui al-Qur'an dan Rasul Allah melalui as-Sunnah yang dipraktikkan oleh jutaan bahkan miliaran umat muslim di mana pun dan kapan pun. Meski demikian, perhatian kaum muslim terhadap kedua sumber pokok ajaran agama masih diabaikan oleh beberapa kalangan yang hanya

mengaku sebagai muslim, sehingga cara beragama sebagian kaum muslim masih terkungkung dalam formalitas yang hanya menjadi rutinitas tanpa makna. Hal ini diperparah pula dengan adanya kejumudan dalam metode pendidikan yang diterapkan, baik dalam lingkungan pendidikan formal maupun informal. Menurut penulis, hal tersebut wajar, mengingat adanya perbedaan jumlah kekayaan wawasan mereka tentang sumber ajaran agama, mulai dari pemahaman teks, hingga penerapan hukum berdasarkan perbedaan pola pikir yang berkaitan dengannya. Oleh karena itu, praktisi pendidikan Islam hendaknya memperhatikan pola pengajaran yang sesuai dengan kondisi peserta didik dan beban ajar yang sanggup dipikul oleh pendidik itu sendiri.

Salah satu aspek pendidikan yang sangat dibutuhkan pada masa ini adalah penguasaan teknologi informasi untuk menunjang hajat hidup masyarakat secara umum. Islam memperhatikan segala aspek yang mencakup kebutuhan seseorang mulai dari kebutuhan primer hingga tersier, tak terkecuali pendidikan terkait teknologi. Mengingat salah satu sumber pokok ajaran Islam yang harus dijunjung tinggi adalah hadits, dan metode menghafal al-Qur'an telah banyak dirumuskan dan diterapkan di berbagai literatur, penulis membatasi penelitian ini hanya dalam koridor hadits, tidak mencakup al-Qur'an. Agar tidak salah paham, penulisan proposal ini tidak bertujuan meneliti kualitas hadits, tidak pula menjelaskan bagaimana pola ajar yang baik terkait kajian al-Qur'an dan Hadits. Itu karena kedua tujuan tersebut berada di luar bidang yang saat ini dikuasai penulis. Fokus penulis

dalam proposal ini adalah membuktikan konsep ilmu komputer, tepatnya pengenalan ucapan dalam membantu menghafal hadits. Tujuan ini sesuai dengan bidang yang hendak ditekuni penulis.

Terkait dengan jumlah Hadits, penulis menggarisbawahi bahwa tidak semua hadits Rasul Allah Muhammad (yang menjadi bahan penelitian penulis) akan dipaparkan dalam makalah ini, karena jumlah hadits dalam kitab-kitab hadits mencapai ribuan, bahkan jutaan. Itu belum termasuk *syarah* (penjelasan) yang ditulis oleh ulama hadits tentang kualitas hadits, baik dari segi sanad (jalur periwayatan) maupun *matan*-nya (isi hadits). Belum termasuk pula sebab-sebab munculnya hadits yang dilatarbelakangi oleh suatu peristiwa berdasarkan data yang ditelusuri oleh para ahli sejarah Islam dan para pakar hadits sendiri. Itu semua membutuhkan penelitian yang membutuhkan waktu lama, tidak ditempuh langsung dalam hitungan hari, tidak pula dalam hitungan minggu maupun bulan. Oleh karena itu, penulis hanya mengkhususkan hadits pilihan yang bersumber dari kitab-kitab tertentu dan telah dinilah *shahih* (memiliki kebenaran yang kuat) dan *hasan* (memiliki kebenaran yang baik) oleh para ulama Hadits.

Berbagai penelitian terkait pengenalan ucapan telah dilakukan oleh beberapa pakar pengolahan suara, terutama dalam Bahasa Arab. Di antaranya adalah pengantar pengenalan ucapan berbahasa Arab menggunakan CMUSphinx (H. Satori, et. al; tanpa tahun), dikembangkan lagi menjadi

implementasi pengenalan ucapan untuk perintah pembacaan al-Qur'an (Yacine Yekache , et. al.; 2011)

H. Satori dkk memperkenalkan penerapan pengenalan ucapan Berbahasa Arab dengan menggunakan CMUSphinx, dengan batasan masalah berupa pengucapan 10 angka Arab (nol hingga sembilan) yang diucapkan masing-masing 5 kali untuk setiap digit oleh 6 orang lelaki asal Maroko, sehingga didapat 300 kali percobaan secara keseluruhan. Hasilnya adalah; dari tiga orang yang diperintahkan mengucapkan 10 angka tersebut dalam Bahasa Arab, dua di antaranya memperoleh persentasi perbandingan pengenalan rata-rata (Mean Recognition Ratio/MRR) 86,66 % dan seorang mendapat nilai MRR 83,33 %.

Penulis menekankan, urgensi Bahasa Arab dalam perkembangan agama Islam merupakan hal yang mutlak diperlukan karena Bahasa Arab dapat menampung makna al-Qur'an secara keseluruhan, sehingga Bahasa Arab dipilih sebagai Bahasa Al-Qur'an. Di samping al-Qur'an, hadits juga menjadi rujukan kedua setelah al-Qur'an karena tiga hal, yaitu

1. Sebagai perinci syari'at yang diwahyukan dalam al-Qur'an (contoh: sholat, zakat, puasa, haji)
2. Sebagai penegasan perintah yang sudah ada dalam al-Qur'an
3. Sebagai penjelasan tambahan terkait hal yang tidak dibahas dalam al-Qur'an

4. Sebagai syari'at yang berdiri sendiri di saat al-Qur'an sama sekali tidak membahasnya

Sistem pengolahan suara menjadi salah satu topik yang terus dibahas dalam disiplin ilmu komputer. Mulai dari operasi sinyal seperti penjumlahan, perkalian, pengurangan, pembagian, pemampatan, penguatan, pelemahan, dan seterusnya. Salah satu sub tema yang kerap dibicarakan adalah pengenalan ucapan (speech recognition).

Sistem pengenalan ucapan sendiri telah banyak melibatkan disiplin ilmu, seperti fisika bunyi, matematika, dan algoritma lain yang diterapkan melalui komputasi. Salah satu metode yang paling dikenal dalam penerapan sistem tersebut adalah Hidden Markov Model (HMM). Metode ini telah banyak diulas dalam berbagai jurnal komputasi. Salah satunya melalui jurnal IEEE (Lawrence R. Rabiner, 1989). Meski pengetahuan dasar tentang HMM telah dikenal sejak tahun 1970-an, pengetahuan metode HMM secara menyeluruh dan penerapannya baru dikenal pada masa-masa kini. Ini dikarenakan pengetahuan dasar tentang HMM diterbitkan dalam jurnal matematika, tidak diketahui umum oleh insinyur yang bekerja dalam masalah pengenalan ucapan. Selain itu, penerapan teori pengenalan ucapan yang sesungguhnya tidak menyediakan materi pelatihan cukup bagi kebanyakan pembaca yang mengerti teori dan sanggup menerapkannya pada bidang penelitiannya sendiri.

Penulis menyadari, hal tersebut memang masih relatif baru untuk generasi masa kini, apalagi terkait dengan praktik yang akan penulis jabarkan dalam tugas akhir ini. Di samping itu, penulis memahami bahwa keahlian ini memang tidak sembarang dapat dikuasai secara menyeluruh oleh seorang ilmuwan komputer sendiri. Sehingga wajar bila penelitian terkait pengenalan ucapan memang melibatkan banyak pakar dari berbagai disiplin ilmu. Oleh karena itu, penulis sengaja membatasi cakupan penelitian tentang pengenalan ucapan, sehingga pembahasan tema yang kurang relevan bagi penulis, seperti analisis data di luar cakupan ilmu komputer (perhitungan gelombang suara, piranti pendukung yang memengaruhi proses pengolahan data) diharapkan dapat diminimalkan.

Terkait implementasi, penulis membatasinya pada kasus pelafalan rangkaian kata dalam hadits, karena hadits tidak begitu membutuhkan cara baca yang sesuai kaidah tajwid dan tartil sebagaimana al-Qur'an. Karena itu, konsep tersebut menginspirasi penulis melakukan penelitian dengan judul *Rancang Bangun Aplikasi Pengenalan Ucapan untuk Kata dalam Hadits Berbahasa Arab Menggunakan Sphinx.*

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah penulis kemukakan, terdapat rumusan masalah berikut

1. Bagaimana rancang bangun aplikasi pelafalan hadits berbasis pengenalan ucapan menggunakan *framework sphinx*?

1.3. Tujuan Penelitian

Bertolak dari rumusan masalah yang telah penulis buat, Penulis memaparkan tujuan penelitian berikut ini

1. Merancang bangun aplikasi pelafalan hadits berbasis pengenalan ucapan menggunakan *framework sphinx*

1.4. Batasan Masalah

Agar pembahasan tidak melebar kepada hal yang di luar konteks penulisan proposal, penulis mempertimbangkan hal-hal berikut terkait pembatasan masalah yang akan dibahas di sini.

1. Hadits yang dijadikan bahan penelitian dalam proposal ini hanya 5 hadits pilihan
2. Hadits yang menjadi objek uji coba data hanya berupa perkataan yang disandarkan kepada Rasul tanpa menyertakan periwayat Hadits beserta jalurnya.
3. Pelafal hadits harus dapat memahami Bahasa Arab dan mengucapkannya dengan benar, karena Bahasa induk hadits adalah Bahasa Arab.

4. Uji coba pengenalan ucapan yang dibahas dalam proposal ini tidak digolongkan berdasarkan dialeknya, sehingga dialek pengucapan Bahasa Arab diabaikan

BAB II

TINJAUAN PUSTAKA

Bab ini dibagi menjadi dua: penelitian terkait yang menjadi rujukan penulis dalam merumuskan penelitian, dan tinjauan pustaka yang membahas istilah dan hak-hal yang terkait dengannya

2.1. Penelitian Terkait

H. Satori dkk (tanpa tahun) memperkenalkan penerapan pengenalan ucapan Berbahasa Arab dengan menggunakan CMUSphinx, dengan batasan masalah berupa pengucapan 10 angka Arab (nol hingga sembilan) yang diucapkan masing-masing 5 kali untuk setiap digit oleh 6 orang lelaki asal Maroko, sehingga didapat 300 kali percobaan secara keseluruhan.

Yacine dkk (2011) memaparkan penelitiannya lebih spesifik dalam mengimplementasikan Bahasa Arab untuk perintah pembacaan al-Qur'an, menggunakan framework Sphinx. Peneliti tidak memaparkan hasil akhir rancang bangun aplikasi tersebut karena penelitian tersebut masih merupakan langkah awal dalam mengembangkan aplikasi pembacaan al-Qur'an yang diperintah melalui suara.

Sebagaimana telah dibahas pada bab lalu, aplikasi tersebut menerapkan sistem pengolahan suara terotomatisasi (Automated Speech Recognition) menggunakan Sphinx dengan memperhatikan tiga kriteria berikut

1. Acoustic Model, yang mewakili jangkauan representasi suara secara statistik untuk mengetahui bentuk fonem
2. Pronunciation Dictionary, yaitu kamus bahasa yang dipakai untuk memetakan cara pengucapan suatu kata menurut fonem yang dipakai dalam Acoustic Model
3. Language Model yang memetakan pola penggunaan kata, namun pola tersebut harus tetap berada dalam Pronounciation Dictionary

2.2. Tinjauan Pustaka

2.2.1. Mel Frequency Cepstral Coefficients

Proses pertama yang dilalui dalam pengolahan suara (speech recognition) adalah ekstraksi fitur. Salah satunya dengan menggunakan metode MFCC. Metode ini diperkenalkan oleh Davis dan Mermelstein di tahun 1980-an. Ekstraksi fitur dalam proses ini ditandai dengan perubahan data suara menjadi data citra berupa spektrum gelombang. Tentu saja, proses ini tidak tampak di permukaan begitu saja, karena proses ini membutuhkan komputasi yang rumit dan hanya dilakukan oleh mesin secara tersembunyi.

Secara sederhana, penulis akan memaparkan langkah-langkah pencarian MFCC dengan asumsi bahwa sinyal suara yang diolah memiliki frekuensi sebesar 16 kHz.

1. Pre-emphasis: Proses ini mencakup penambahan energi suara pada frekuensi tinggi. Secara matematis, dapat dirumuskan dalam bentuk berikut

$$y(n) = x(n) – 0.95\ x(n − 1)$$

di mana y(n) adalah sinyal yang ditekankan, sedangkan x(n) adalah sinyal terdigitasi. Koefisien dengan nilai 0.95 menunjukkan sinyal yang diekstrak merupakan 95 % sinyal aslinya.

2. Framing: Susun sinyal ke dalam bingkai yang lebih pendek. Panjang frame yang membagi setiap sample menjadi beberapa frame berdasarkan waktu terletak di antara 20 hingga 40 ms, standarnya adalah 25 ms. Dengan asumsi bahwa frekuensi suara 16 kHz, maka sampel yang akan diekstrak adalah 0,025 detik * 16.000 hZ = 400 sampel.
3. Windowing: Jendela Hamming digunakan seperti bentuk jendela dengan mempertimbangkan blok berikutnya dalam rantai pemrosesan ekstraksi fitur dan memadukan semua garis frekuensi terdekat. Persamaan jendela Hamming adalah sebagai berikut:

$$w(n)=0.54-0.46\cos\left[\frac{2\cdot\pi\cdot n}{N-1}\right],0\le n\le N-1$$

Setelah itu, kalikan hasil persamaan jendela Hamming dengan sinyal yang telah ditekankan

$$Y(n) = y(n) * w(n)$$

di mana

- N = Banyaknya sampel tiap frame
- Y(n) = Sinyal Output
- y(n) = Sinyal Input
- w(n) = Jendela Hamming

4. Fast Fourier Transform: Langkah ini berguna untuk mengubah tiap frame N sampel dari domain waktu ke dalam domain frekuensi. Dalam pengolahan suara, Transformasi Fourier Cepat berguna untuk mengubah konvolusi getaran celah suara dan respon gelombang saluran suara dalam domain waktu. Rumus Transformasi Fourier Diskrit J(k) dapat dilihat di bawah ini

$$J(k) = \sum_{k=0}^{N-1} [Y(n) e^{-\frac{j \cdot 2 \cdot \pi \cdot k \cdot n}{N}}]; k = 0, 1, \ldots, N-1$$

5. Mel Filterbank Processing: Jangkauan frekuensi dalam spektrum sangatlah luas dan sinyal suara tidak mengikuti skala linear. Sehingga setelah spektrum terkomputasi, data dipetakan dalam skala Mel menggunakan filter segitiga yang saling tumpang tindih dengan empat langkah berikut.
   1. Tentukan banyaknya mel bank filter m di mana m = 1, 2, 3, …, M adalah banyaknya bank filter

2. Hitung frekuensi pusat. Tiap besarnya respon frekuensi filter berbentuk segitiga dan sama dengan satu kesatuan pada frekuensi pusat dan berkurang secara linear menuju nol pada pusat frekuensi dua filter yang bersebelahan. Frekuensi pusat dirumuskan sebagai

   φ c = m. Δφ c, di mana

$$\Delta\varphi c = \frac{(f_{max})_{(Mel)} - (f_{min})_{(Mel)}}{M+1}(Mel)$$

   di mana $(f_{max})_{(Mel)}$ dan $(f_{min})_{(Mel)}$ adalah kecocokan skala hertz dalam skala Mel. Kecocokan tersebut dihitung dengan rumus berikut

$$f(mel) = 2595 \log_{10}\left(\frac{f(hertz)}{700} + 1\right)$$

3. Hitung mel-filterbank. Ini dipakai untuk menyaring ceptrum utama. Didapat dengan menggunakan frekuensi cepstral yang terkonversi (mel ke Hz), yang dirumuskan secara matematis sebagai berikut

$$R(k,m) = \begin{cases} 0, f(k) < f_c(m-1) \\ \frac{f(k) - f_c(m-1)}{f_c(m) - f_c(m-1)}, f_c(m-1) \le f(k) < f_c(m+1) \\ \frac{f(k) - f_c(m+1)}{f_c(m) - f_c(m+1)}, f_c(m-1) \le f(k) < f_c(m+1) \\ 0, f(k) \ge f_c(m+1) \end{cases}$$

   di mana skala hertz $f_c(m)$ yang cocok dengan skala mel φ_c dihitung dengan rumus;

$$f_c(m) = 700\left(10^{\frac{f(mel)}{2595}} - 1\right)$$

4. Petakan frekuensi dari Hertz ke skala Mel. Ini merupakan langkah terakhir pemrosesan Mel-Filterbank. Cara ini dapat ditempuh dengan rumus

$$J_1(m) = \sum_{k=0}^{N-1} |J(k)| R(k, m)$$

di mana, J_1 (m) adalah Mel spectrum.

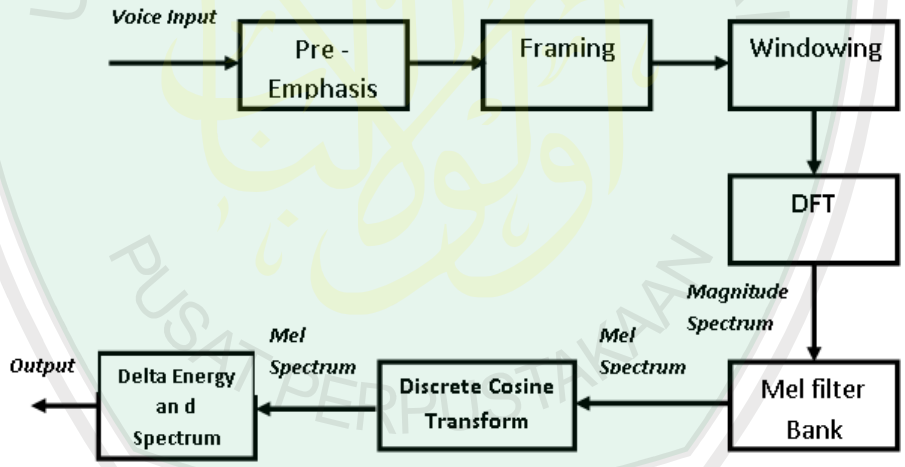
6. Discrete Cosine Transform: Dipakai untuk mencari nilai koefisien Mel cepstrum. Ini didapat dengan menghitung logaritma Mel Cepstrum, kemudian ditransformasikan ke dalam domain frekuensi.
7. Delta Energy dan Delta Spectrum: Sinyal suara dan frame berubah bentuk seperti lereng pada pergantian fasenya. Sehingga, perlu menambahkan fitur yang terkait dengan perubahan fitur cepstral tiap waktu. Tiga belas delta atau fitur kecepatan (12 fitur cepstral ditambah energi), dan 39 fitur double delta atau fitur percepatan ditambahkan Energi di dalam frame untuk sebuah sinyal x dalam jendela dari sampel waktu t1 hingga sampel waktu t2, yang diwakili oleh persamaan berikut.

$$Energy = \sum X^2[t]$$

Tiap 13 fitur delta mewakili perubahan antar frame dalam cepstral atau fitur energi yang cocok, sedangkan tiap 39 fitur double delta mewakili perbubahan antar frame dalam fitur delta yang sesuai.

$$d(t)=\frac{c(t+1)-c(t-1)}{2}$$

Penjabaran ringkas terkait prosedur yang dijalankan dalam ekstraksi fitur dengan metode MFCC dijabarkan dalam gambar 2.1. Hasil ekstraksi fitur ini yang akan digunakan sebagai input untuk pemodelan dengan metode Hidden Markov Model.



Gambar 2.1: Alur proses ekstraksi fitur menggunakan metode MFCC (Sumber: practicalcryptography.com)

2.2.2. Hidden Markov Model

Hidden Markov Model merupakan suatu model matematika berbasis peluang yang didasarkan pada parameter yang tidak

diketahui secara pasti oleh komputer. Model ini kerap digunakan sebagai metode untuk melakukan pengenalan pola pada suara sehingga dapat diubah menjadi teks. Ada beberapa hal yang dibutuhkan dalam menerapkan metode ini

1. Topologi model, mencakup jumlah kondisi dan hubungan antarkondisi yang memungkinkan
2. Peluang perpindahan keadaan, juga mencakup fungsi keadaan berikutnya, dengan rumus $P(q(t + 1) = m \mid q(t) = l)$
3. Peluang keluaran, dapat digambarkan dengan sebuah histogram untuk setiap keadaan. Jika terdapat simbol output R ($O = \{O_1, O_2, \ldots, O_{R-1}, O_R\}$), maka setiap keadaan $q(t) = l$ akan menghasilkan sebuah histogram peluang R, berupa $P(O(t) = i \mid q(t) = l)$

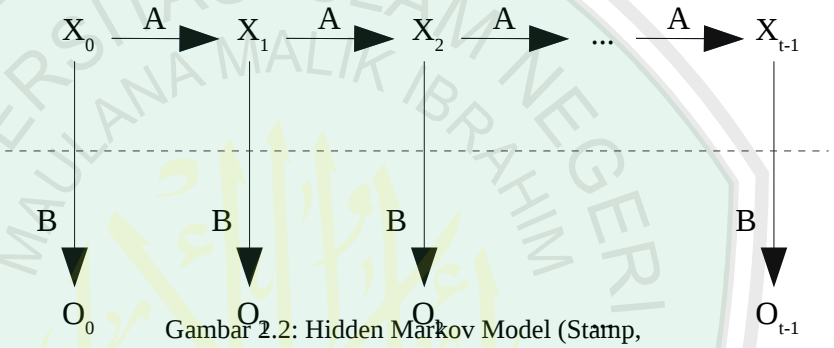
Dua himpunan peluang (sebagaimana disebutkan di atas) merupakan parameter model, yang dilambangkan dengan simbol X. Sedangkan model itu sendiri dilambangkan dengan simbol λ. Penulis akan menggunakan notasi $O = (O(1), O(2), \ldots, O(N - 1), O(N))$ untuk mewakili serangkaian simbol output., dan notasi $Q_1^N = (q(1), q(2), \ldots, q(N - 1), q(N))$ untuk mewakili rangkaian keadaan.

Selanjutnya, topologi model harus ditentukan secara rinci. Setelah itu, ada tiga masalah yang akan dikemukakan pada saat

implemetasi metode ini ke dalam pengenalan ucapan (Renals, 1996).

1. Problem Evaluasi, mencakup cara menghitung peluang P (O | X, λ). Artinya dengan memasukkan parameter X dan λ akan menghasilkan output O. Solusi permasalahan ini memungkinkan pencarian peluang perbedaan HMM yang menghasilkan kesamaan rangkaian pengamatan. Komputasi ini dapat dipakai untuk membandingkan peluang perbedaan model yang menghasilkan kesamaan data. Jika ada berbagai perbedaan HMM untuk setiap kata, maka kata yang dikenal merupakan kata yang modelnya memiliki peluang pembentukan kata terbesar.
2. Problem Decoding, setelah menentukan paramter model, model itu sendiri, dan serangkaian output, selanjutnya menghitung peluang rangkaian keadaan terbesar Q_1^N yaitu $Q_1^N = \text{argmax}_R P (R_1^N | O, X, \lambda)$. Masalah ini diselesaikan dengan mengungkap rangkaian keadaan tersembunyi dari bekal pengetahuan simbol output oleh mesin. Solusi permasalahan ini berguna untuk mendapatkan maksud keadaan machine yang luar biasa. Ia juga menjadi solusi yang penting dalam kebanyakan aplikasi HMM saat model dasar disatukan bersama.

3. Problem Training, meliputi cara mengatur parameter model untuk memperbesar peluang model λ yang memproduksi rangkaian output O. Solusi permasalahan ini akan menyajikan cara otomatis untuk memperkirakan parameter (peluang perpindahan dan output) setiap HMM, menggunakan himpunan data training.



Gambar 2.2: Hidden Markov Model (Stamp, 2012)

Konsep sederhana Hidden Markov Model dapat dilihat pada gambar 2.2, Simbol X_i mewakili rangkaian keadaan tersembunyi dan semua notasi lain dapat dijelaskan secara singkat dalam tabel dengan keterangan sebagai berikut

Simbol	Penjelasan
A	Matriks peluang perpindahan peadaan
B	Matriks peluang pengamatan
M	Jumlah simbol pengamatan
N	Jumlah keadaan dalam Model
O	Rangkaian Observasi
T	Banyaknya rangkaian Observasi
Q	Keadaan Proses Markov yang Tampak
V	Himpunan Pengamatan yang Mungkin

Simbol	Penjelasan
X	Rangkaian Keadaan yang Tersembunyi
π	Persebaran keadaan awal

Proses Markov (yang terletak di belakang garis putus-putus) ditentukan oleh keadaan terkini dan matriks A. Kita hanya sanggup mengamati rangkaian observasi O_i, yang dihubungkan dengan keadaan proses Markov tersembunyi oleh matriks B.

Matriks A = $\{a_{ij}\}$ memiliki ordo N × N dengan

$$a_{ij} = P(keadaan\, q_j\, pada\, waktu\, (t+1) | keadaan\, q_i\, pada\, waktu\, t)$$

Matriks B= $\{b_j(k)\}$ memiliki ordo N × M dengan

$$b_j(k) = P(pengamatan\, k\, pada\, waktu\, t | keadaan\, q_j\, pada\, waktu\, t).$$

Kedua matriks tersebut adalah matriks stokastik di setiap barisnya, dengan jumlah elemen pada masing-masing baris bernilai 1. Lalu besar peluang pada tiap elemen matriks (baik a_{ij} maupun $b_j(k)$) bersifat independen terhadap waktu t. Sedangkan π_{x0} adalah peluang permulaan pada keadaan x_0 . Juga, $b_{x0}(O_0)$ peluang awal pada pengamatan objek O_0 dan $a_{x0,x1}$ adalah peluang perpindahan dari keadaan x_0 menuju keadaan x_1.

Misalkan ada serangkaian keadaan umum X dirumuskan dalam panjang t, berarti

$$X = (x_0, x_1, x_2, \ldots, x_{(t-1)})$$

Sedangkan serangkaian pengamatan O dirumuskan dalam panjang t, berarti

$$O = (O_0, O_1, O_2, \ldots, O_{(t-1)})$$

Dengan demikian, peluang keadaan X dapat dicari dengan rumus berikut

$$P(X)=\pi_{x_0} b_{x_0}(O_0)\, a_{x_0,x_1} b_{x_1}(O_1)\, a_{x_1,x_2} b_{x_2}(O_2)\, a_{x_2,x_3} b_{x_3}(O_3) \ldots a_{x_{(t-2)},x_{(t-1)}} b_{x_{(t-1)}}(O_{(t-1)})$$

Secara ringkas, Hidden Markov Model dapat dirumuskan sebagai berikut

$$\lambda=(A, B, \pi)$$

Simbol-simbol di atas tidak hanya mewakili Gambar 2.2, namun juga akan diterapkan pada penjabaran matematis dari Hidden Markov Model itu sendiri, tepatnya ketiga permasalahan yang telah dipaparkan di muka, yakni sebagai berikut

Berikut ini merupakan penjelasan ringkas yang mewakili solusi ketiga permasalahan yang telah dikemukakan di muka. (Stamp, 2012)

1. Pilih nilai asal untuk matriks A, B dan π, di mana π adalah matriks berordo $1 \times N$, sedangkan $A = \{a_{ij}\}$ adalah matriks berordo $N \times N$ dan $B = \{b_j(k)\}$ adalah matriks berordo $N \times M$, dan ketiganya memiliki nilai yang disebar secara acak berdasarkan baris. Jika telah diketahui, gunakan pendekatan estimasi untuk nilai matriks, jika tidak, anggap

$\pi_i \approx 1 / N$ dan $a_{ij} \approx 1 / N$ dan $b_j(k) \approx 1 / M$. Pastikan bahwa tiap baris matriks berjumlah 1 dan unsur setiap matriks tidak seragam. Anggap

1. maxIters = jumlah iterasi maksimum,
2. iters = 0, dan
3. oldLogProb = $-\infty$.

2. Algoritma α-pass

```
// hitung α0(i)
c0 = 0
for i = 0 to N − 1
        α0(i) = πi bi(O0 )
        c0 = c0 + α0(i)
next I

// atur skala α0(i)
c0 = 1 / c0
for i = 0 to N − 1
        α0(i) = c0 α0(i)
next I

// hitung αt (i)
for t = 1 to T − 1
        ct = 0
        for i = 0 to N − 1
                αt(i) = 0
                for j = 0 to N − 1
                        αt(i) = αt(i) + αt−1(j) aji
                next j
                αt(i) = αt(i) bi(Ot)
                ct = ct + αt(i)
        next I

        // atur skala αt(i)
        ct = 1 / ct
        for i = 0 to N − 1
                αt(i) = ct αt(i)
        next i
next t
```

3. Algoritma β-pass

```
// Anggap β_{T-1}(i) = 1 diatur skalanya oleh c_{T-1}
for i = 0 to N − 1
        β_{T-1}(i) = c_{T-1}
next I

// β-pass
for t = T − 2 to 0 by − 1
        for i = 0 to N − 1
                β_t(i) = 0
                for j = 0 to N − 1
                        β_t(i) = β_t(i) + a_ij b_j(O_{t+1}) β_{t+1}(j)
                next j

                // Atur skala β_t(i) denga faktor skala sama dengan α_t(i)
                β_t(i) = c_t β_t(i)
        next i
next t
```

4. Hitung $\gamma_t(i, j)$ dan $\gamma_t(i)$

```
for t = 0 to T − 2
        denom = 0
        for i = 0 to N − 1
                for j = 0 to N − 1
                        denom = denom + α_t(i) a_ij b_j(O_{t+1}) β_{t+1}(j)
                next j
        next i
        for i = 0 to N − 1
                γ_t(i) = 0
                for j = 0 to N − 1
                        γ_t(i, j) = (α_t(i) a_ij b_j(O_{t+1})β_{t+1} (j)) / denom
                        γ_t(i) = γ_t(i) + γ_t (i, j)
                next j
        next i
next t
```

5. Re-estimasi A, B and π

```
// re-estimasi π
for i = 0 to N − 1
        πi = γ0(i)
next I

// re-estimasi A
for i = 0 to N − 1
        for j = 0 to N − 1
                numer = 0
                denom = 0
                for t = 0 to T − 2
                        numer = numer + γt (i, j)
                        denom = denom + γt (i)
                next t
                aij = numer / denom
        next j
next I

// re-estimate B
for i = 0 to N − 1
        for j = 0 to M − 1
                numer = 0
                denom = 0
                for t = 0 to T − 2
                        if(O t == j) then
                                numer = numer + γt (i)
                        end if
                        denom = denom + γt (i)
                next t
        bi (j) = numer / denom
        next j
next i
```

6. Hitung log[P (O | λ)]

```
logProb = 0
for i = 0 to T − 1
        logProb = logProb + log(ci)
next i
logProb = −logProb
```

7. Tentukan apakah iterasi berjalan atau tidak

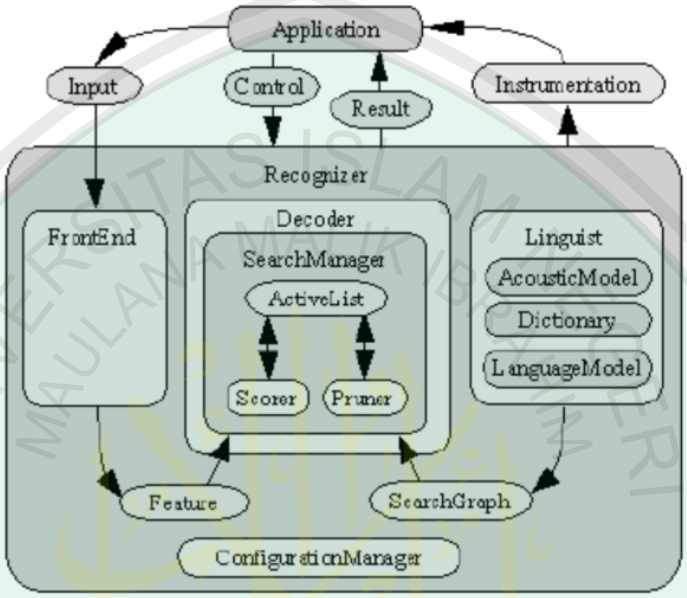
```
iters = iters + 1
if (iters < maxIters and logProb > oldLogProb) then
        oldLogProb = logProb
        goto step 2
else
        output λ = (π, A, B)
end if
```

2.2.3. Sphinx-4

Sphinx-4 adalah sebuah library yang digunakan untuk pengolahan suara. Library ini disusun oleh tim gabungan dari Universitas Carnegie Mellon, Laboratorium Penelitian Listrik Mitsubishi, dan Sun Microsystem.

Dalam penelitian yang dilakukan oleh Willie Walker (2004), framework ini diciptakan dengan latar belakang adanya kesulitan dalam melakukan konfigurasi parameter karena framework yang ada sangat bergantung sepenuhnya kepada sistem yang berlaku, sehingga pola desain yang berlaku bersifat kaku. Atas dasar itu, peneliti mencoba merancang bangun framework yang dapat dikonfigurasi untuk setiap percobaan yang berbeda. Dalam mengembangkan Sphinx-4, salah satu tujuan yang akan dicapai tidak terbatas pada pengembangan sebuah framework yang mendukung pola desain pengenalan ucapan pada bahasa tertentu, tetapi juga memungkinkan percobaan dalam lingkup penelitian yang berbeda. (Willie Walker , et. al., 2004)

Sphinx memiliki tiga blok utama dalam menangani pengenalan ucapan, yaitu FrontEnd, Linguist, dan Decoder. Secara umum, model Arsitektur Sphinx ditunjukkan pada gambar 2.3.

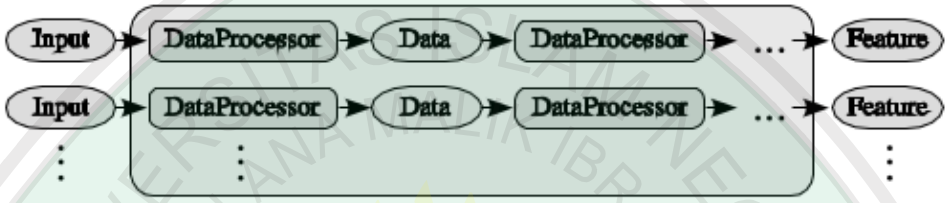


Gambar 2.3: Struktur Framework Sphinx (Walker , et. al., 2004)

Berikut ini merupakan gambaran ringkas masing-masing blok framework beserta penjelasan terkait mekanisme kerjanya.

1. FrontEnd: Blok ini bertanggung jawab dalam menangani suara input sehingga dapat terekstrak menjadi berbagai fitur. Sebagaimana dijelaskan dalam gambar 2.4, FrontEnd mengandung satu atau lebih rantai modul pemrosesan sinyal komunikasi yang dapat dipindahkan, dinamakan DataProcessor. Dukungan banyak rantai mengizinkan komputasi serentak perbedaan tipe parameter dari sinyal

input yang sama atau berbeda. Ini memungkinkan pembuatan sistem yang dapat mengurai kode secara serentak menggunakan tipe parameter yang berbeda, seperti MFCC, PLP, dan bahkan tipe parameter yang diturunkan dari sinyal non-suara seperti video.



Gambar 2.4: Struktur blok FrontEnd (Walker , et. al., 2004)

Dalam framework FrontEnd umum, Sphinx-4 menyediakan serangkaian DataProcessor yang menerapkan teknik pemrosesan sinyal pada umumnya. Penerapan ini mencakup dukungan untuk langkah berikut: Membaca dari sebuah variasi format input untuk operasi mode batch, membaca dari sistem piranti input audio untuk operasi mode langsung (live), pra emphasis, windowing dengan Transformasi Kosinus tinggi (raised cosine transform, contohnya Hamming and Hanning windows), Transformasi Fourier Diskrit (via FFT), penyaringan frekuensi mel, pelengkungan frekuensi tepi, transformasi kosinus diskrit (DCT), Pengkodean Prediktif Linear (LPC, Linear Predictive Coding), penunjukan titik akhir (end pointing),

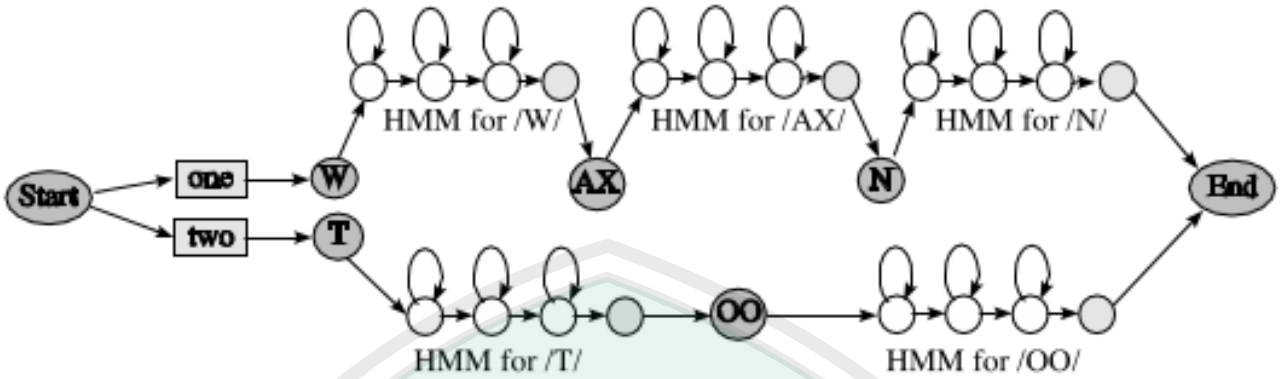
normalisasi Cepstral rata-rata (CMN, Cepstral Mean Normalization), ekstraksi koefisien cepstra frekuensi mel (MFCC, Mel Frequency Cepstral Coefficients), dan ekstraksi koefisien prediksi linear persepsi. extraction (PLP, Perceptual Linear Prediction Coefficient).

2. Linguist: Blok ini berperan dalam menentukan kaidah/aturan bahasa yang diterapkan terkait fitur suara yang telah diekstrak. Blok ini terdiri dari tiga komponen
    1. Acoustic Model: Modul ini menyediakan sebuah pemetaan antara satuan pengucapan dan sebuah HMM yang dapat dinilai terhadap fitur yang disediakan oleh FrontEnd. Secara khusus, blok Linguist memecah tiap kata yang terdapat di dalam kosa kata aktif menjadi serangkaian satuan suku kata bergantung konteks. Lalu Linguist melewati satuan kata tersebut dan konteksnya menuju *Acoustic Model*, menerima graf HMM terkait satuan-satuan tersebut. Kemudian menggunakan graf HMM ini bersamaan dengan *Language Model* untuk membentuk *SearchGraph*.
    2. Language Model: Modul ini menyediakan struktur bahasa tingkat kata, yang dapat diwakili oleh sejumlah implementasi yang dapat dipasang. Implementasi ini

secara jenisnya dibagi menjadi dua kategori: tata bahasa yang didorong oleh graf dan model N-Gram stokastik. Tata bahasa yang dikendalikan oleh graf mewakili sebuah graf kata di mana tiap simpul (node) mewakili sebuah kata tunggal dan tiap busur mewakili kemungkinan perubahan kata yang terjadi. Model N-Gram Stokastik menyediakan kemungkinan kata yang diberikan berdasarkan pengamatan n-1 kata sebelumnya.

3. Dictionary: Modul ini menyediakan pengucapan tiap kata yang terdapat dalam Language Model. Pengucapan kata (Word Recognizer) memecah kata menjadi serangkaian satuan suku kata yang ditemukan dalam Acoustic Model. Antar muka Dictionary juga mendukukng pengelompokan kata dan memungkinkan sebuah kata tunggal menjadi banyak kelas.

Satu lagi komponen yang berperan, yaitu *SearchGraph* yang menjadi penghubung antara blok Decoder dan Linguist (lihat kembali gambar 2.3). Komponen ini berfungsi memetakan cara pencarian dan topologi ruang pencarian. Contoh penerapan *SearchGraph* dapat dilihat pada gambar 2.5.



Gambar 2.5: Contoh penerapan komponen *SearchGraph* (Walker , et. al., 2004)

Pola pencarian yang diilustrasikan pada gambar 2.5 menunjukkan bahwa *SearchGraph* berbentuk graf terarah yang terdiri dari *SearchState*s yang muncul beserta *SearchStateArcs* yang keluar secara opsional dengan nilai peluang perpindahan. Tiap keadaan dalam graf dapat mewakili komponen Language Model (kata dalam persegi panjang), Dictionary (satuan suku kata di dalam lingkaran) atau Acoustic Model (HMM).

3. Decoder: Blok ini berfungsi sebagai penerjemah berbagai fitur ke dalam teks sesuai dengan kaidah bahasa yang dipakai dalam blok linguist, sehingga output yang dihasilkan tidak keluar dari aturan yang berlaku. Peran utama Blok Decoder pada Sphinx-4 adalah memanfaatkan Feature dari blok FrontEnd bersamaan dengan *SearchGraph* dari Linguist untuk menghasilkan Hipotesa

Hasil. Blok Decoder mengandung sebuah *SearchManager* yang dapat dipasang dan kode pendukung yang menyederhanakan proses decoding untuk sebuah aplikasi. Dengan demikian, komponen blok Decoder yang paling menarik adalah *SearchManager*.

Tugas Decoder hanyalah memberitahukan *SearchManager* terkait pengenalan sebuah himpunan bingkai fitur. Di setiap langkah proses, *SearchManager* membuat objek hasil yang mengandung semua jalur yang telah mencapai keadaan akhir yang tak keluar. Untuk memproses hasil, Sphinx-4 juga menyediakan kegunaan yang cakap dalam memproduksi sebuah kisi dan skor rahasia dari blok *Result*. Akan tetapi, tidak seperti sistem lain, aplikasi yang menerapkan framework ini memungkinkan perubahan ruang dan objek hasil di antara beberapa langkah, mengizinkan aplikasi menjadi pasangan dalam proses pengenalan.

Seperti halnya Linguist, SearchManager tidak terbatas pada implementasi khusus. Sebagai contoh, implementasi *SearchManager* dapat melakukan algoritma pencarian seperti frame-synchronous Viterbi, A*, bi-directional, dan lain-lain.

Tiap implementasi *SearchManager* menggunakan algoritma token passing. Token Sphinx-4 adalah sebuah objek yang terkait dengan *SearchState,* mengandung nilai akustik dan bahasa rata-rata dari jalan (path) di titik yang diberikan, sebuah acuan menuju *SearchState*, sebuah acuan menuju sebuah fitur input, dan informasi relevan lainnya. Acuan *SearchState* memungkinkan *SearchManager,* mengaitkan sebuah token dengan distribusi output keadaannya, satuan fonem yang bergantung konteks, pengucapan, kata, dan keadaan tata bahasa. Tiap hipotesis parsial berakhir di sebuah token aktif.

Seperti ditunjukkan dalam gambar 2.3, implementasi *SearchManager* dapat membangun sebuah himpunan token aktif di dalam bentuk *ActiveList* di tiap langkah waktu, meski penggunaan *ActiveList* tidak disyaratkan. Akan tetapi karena itu merupakan teknik yang bersifat umum, Sphinx-4 menyediakan sub-blok untuk mendukung *SearchManager* yang terdiri atas *ActiveList*, *Pruner*, dan *Scorer*.

Sub-blok *SearchManager* menghasilkan *ActiveLists* dari token aktif terkini dalam bingkai pencarian dengan pemangkasan dengan menggunakan sub-framework *Pruner* yang dapat dipasang. Aplikasi dapat mengatur

setelan Implementasi *Pruner* pada Sphinx-4 untuk melakukan pemangkasan lebar, baik relatif maupun absolut. Penerapan Pruner sangat disederhanakan dengan pengumpulan sampah (garbage collector) Java Platform. Dengan pengumpulan sampah, the *Pruner* dapat memangkas jalan dengan hanya menghilangkan token terminal jalan dari *ActiveList.* Tindakan penghilangan token terminal memperkenalkan token dan tiap token yang tak terbagi untuk jalan itu saat tidak dipakai, memungkinkan pengumpul sampah untuk mengambil kembali memori terkait.

Sub-blok *SearchManager* juga berhubungan dengan *Scorer*, sebuah modul perkiraan (estimasi) peluang keadaan terpasang, yang menyediakan nilai kepadatan output keadaan di saat ada permintaan. Saat *SearchManager* meminta sebuah nilai untuk keadaan dan waktu yang diberikan, *Scorer* mengakses vektor fitur pada waktu itu dan melakukan operasi matematika untuk menghitung sebuah nilai. Dalam kasus paralel decoding yang menggunakan model akustik paralel, *Scorer* mencocokkan himpunan model akustik untuk digunakan terhadap tipe fitur.

Scorer menarik semua informasi yang berkaitan dengan kepadatan output keadaan. Dengan demikian, SearchManager tidak perlu lagi mengetahui apakah penilaian dilakukan dengan HMM kontinu, setengah kontinu, atau pun diskrit. Lebih jauh, fungsi kepadatan peluang untuk setiap keadaan HMM diisolir dengan cara yang sama. Algoritma heuristik apapun yang termasuk ke dalam prosedur penilaian untuk melakukan percepatan juga dijalankan secara setempat di dalam *Scorer*. Sebagai tambahan, *scorer* dapat mengambil keuntungan banyak CPU jika tersedia.

Blok tersebut bekerja di balik layar, sehingga proses perubahan suara menjadi teks tidak kasat mata. Lebih lanjut, sifat modular sistem Sphinx-4 utamanya dapat diaktifkan oleh penggunaan bahasa pemrograman Java. Secara khusus, kemampuan platform Java memuat kode yang berada dalam proses berjalan mengizinkan dukungan sederhana untuk framework yang dapat dipasang, dan bahasa pemrograman Java mengizinkan pemisahan desain framework dari penerapannya. Penggunaan platform Java juga menyediakan framework Sphinx-4 dengan sejumlah kelebihan, antara lain:

1. Sphinx-4 dapat berjalan dalam bermacam-macam platform tanpa rekompilasi, sehingga tidak perlu membuat ulang program ketika pindah platform.
2. Himpunan platform API yang kaya dapat mengurangi banyak waktu yang dibutuhkan untuk pengkodean.
3. Dukungan bawaan untuk multithreading membuat desain framework Sphinx-4 terlihat sederhana, terutama dalam eksperimen tugas persebaran penguraian kode terhadap banyak thread.
4. Pengumpulan sampah otomatis membantu pengembang memusatkan pengembangan algoritma, walaupun terjadi kebocoran memori.

2.2.4. Java Speech Grammar Format (JSGF)

Java Speech Grammar Format (JSGF) merupakan salah satu format tata bahasa yang dipakai dalam sistem pengenalan ucapan (Speech Recognition). Terdiri dari serangkaian aturan tata bahasa tekstual, Berkas JSGF dapat diedit, dibaca, dan dimasukkan ke dalam source code. Baik oleh developer, maupun oleh komputer sendiri. Dalam penelitian kali ini, penulis membatasi pembuatan berkas JSGF secara manual dengan kaidah yang akan dijelaskan secara khusus dalam subpoin berikutnya. Namun secara garis besar, penulis

menekankan ketentuan umum terkait penulisan komponen JSGF sebagai berikut.

1. Nama paket (package) dan grammar. Seperti di java, penulisan nama grammar dilakukan dengan dua cara.
   1) Diawali dengan penentuan nama package, yang dipisahkan dengan tanda titik, dan diakhiri dengan nama grammar. Berikut ini merupakan contoh penulisan grammar bentuk pertama.

      com.sun.speech.apps.numbers

   2) Ditulis langsung dengan nama grammar saja. Contoh penulisan tertera dalam bentuk berikut ini.

      numbers

   Bentuk pertama merupakan nama lengkap grammar, sedangkan nama kedua disebut dengan nama grammar sederhana. Sekali lagi, aturan tersebut mengikuti ketentuan penamaan paket dan kelas pada Bahasa Pemrograman Java. Dalam bentuk pertama, penulisan antar paket dipisahkan dengan tanda titik, dan diakhiri dengan nama grammar.
2. Nama aturan (rulename). Grammar terdiri dari berbagai nama aturan baik tunggal maupun majemuk. Penulisannya diapit dengan tanda kurang-lebih dari (<>). Karakter yang diperbolehkan serupa dengan karakter yang dizinkan dalam

penulisan identifier dalam Java, namun boleh juga disertai dengan karakter tambahan. Karakter tambahan yang dimaksud adalah simbol tanda baca seperti +, - : ; , = | / \ ( ) [ ] @ # % ! ^ & ~. Panjang karakter maksimum yang diperbolehkan tidak terbatas sesuai karakter yang dapat dibaca dengan sistem Unicode Namun, penulisan nama aturan bersifat unik dan sensitif terhadap besar-kecil huruf. Karena itu, tidak boleh ada dua atau lebih aturan yang sama dalam satu rangkaian grammar. Di samping itu, nama aturan tidak boleh mengandung unsur spasi, baik di awal, di tengah, maupun di akhir. Dengan demikian, dapat diketahui bahwa nama aturan untuk <nama> berbeda dengan <Nama> dan <NAMA>. Karakter unicode tidak hanya mencakup huruf latin, melainkan mendukung berbagai macam bahasa di dunia seperti Cina, Jepang, Arab, Korea, Thailand, India, dan bahasa lainnya. Oleh sebab itu, penulisan nama aturan berikut diperbolehkan

- <hello>
- <Zürich>
- <user_test>
- <$100>
- <1+2=3>

- <π α β>

Terkait penulisan nama aturan (rulename), ada syarat yang berlaku dalam penggunaan rulename

1) Penulisan rulename baik lengkap maupun sebagian tidak boleh dilakukan di ruas sebelah kiri penentuan sebuah aturan
2) Pernyataan import harus menggunakan rulename yang ditulis secara lengkap
3) Aturan lokal (berada dalam satu rangkaian grammar) dapat diacu oleh nama aturan, baik ditulis secara lengkap maupun ditulis sebagian dengan bentuk <namaGrammarLokal.namaAturan>

Setelah meninjau syarat di atas, berikut ini merupakan hal yang harus diperhatikan dalam menetapkan acuan

1) Aturan lokal (local rule) memiliki awalan. Jika aturan local dan salah satu atau lebih aturan yang diimpor memiliki nama yang sama, katakanlah <nama>, maka aturan sederhana yang merujuk ke <nama> dirujuk kepada aturan lokal.
2) Jika dua atau lebih aturan yang diimpor memiliki nama yang sama, <nama>, namun tidak ada aturan lokal yang memiliki kesamaan nama, maka aturan sederhana yang

mengacu ke <nama> akan bersifat ambigu, sehingga akan dipandang sebagai galat (error). Untuk menangan ambiguitas ini, aturan yang diimpor ini harus dirujuk oleh nama aturan lengkap atau sebagian

3) Jika dua atau lebih aturan yang diimpor memiliki nama yang sama dan berasal dari grammar dengan nama yang sama, maka aturan acuan (baik sebagian maupun lengkap) akan bersifat ambigu sehingga dipandang sebagai galat. Aturan yang diimpor ini harus dirujuk oleh nama aturan lengkapnya.
4) Rujukan nama aturan lengkap tidak boleh bersifat ambigu

Di saat acuan nama aturan tidak dapat ditetapkan (tidak ditentukan atau bukan aturan grammar publik yang diimpor), penanganan acuan akan ditentukan oleh interface software. Di samping itu, JSGF memperkenalkan aturan spesial yang bersifat universal, tersedia dalam grammar mana pun tanpa pernyataan impor dan tidak dapat ditetapkan ulang. Keduanya tidak perlu ditulis ulang secara parsial karena sudah bersifat lengkap. Aturan yang dimaksud adalah

- <NULL>. Aturan ini akan secara otomatis mengisi kekosongan kata dengan aturan yang cocok tanpa pengucapan kata dari pengguna.

- <VOID>. Aturan ini berisi kata-kata yang tidak pernah dapat diucapkan. Artinya, jika aturan ini disisipkan ke dalam sebuah rangkaian kata, maka otomatis aturan yang tersisipkan nama aturan ini tidak akan bisa diucapkan ssluruhnya.

3. Token menjadi bagian grammar yang menentukan input yang akan diucapkan pengguna, baik berupa kata maupun kalimat. Dalam JSGF, token terdiri dari rangkaian karakter yang dipisahkan dengan spasi, tanda kutip, atau dibatasi dengan simbol khusus, seperti ; = | * + <> () [] {} /* */ //. Aturan penggunaan simbol tersebut akan dijelaskan pada bagian lain subpoin ini.

   Kebanyakan pengenal memiliki kosakata dapat dikenali untuk setiap bahasa yang mereka dukung. Namun, tidak mungkin 100% dapat dicakup dalam bahasa yang bersangkutan. Misalnya, nama, istilah teknis, dan kata asing yang sering hilang dari daftar kosakata. Untuk token yang hilang dari daftar kosakata, ada tiga kemungkinan.

   - Aplikasi atau pengguna dapat menambah token dan cara pengucapan ke dalam kosakata recognizer untuk memastikan pengenalan yang konsisten.

- Recognizer yang baik adalah pengenal yang mampu menebak cara pengucapan kata yang tidak berada di dalam daftar kosakata.
- Jika tidak ada poin sebelumnya yang berlaku, perilaku pengenalan akan ditentukan dengan interface software pengenal. Dalam banyak kasus, token yang tidak ditentukan tidak akan dapat diucapkan (setara dengan <VOID>), atau membangkitkan error atau exception.

Sebuah token tidak harus selalu kata. Sebuah token boleh berupa serangkaian kata atau simbol. Tanda kutip dapat dipakai untuk mengelilingi token dengan banyak kata dan simbol khusus. Berikut ini contohnya

Stasiun "Kota Baru"

"*"

Sebuah token multi-kata berguna di saat cara pengucapan kata bermacam-macam, tergantung konteksnya. Token banyak kata juga dapat digunakan untuk menyederhanakan hasil pemrosesan. Contohnya, mengambil token tunggal yang hasilnya hanya “Kota Baru”, sebagaiman contoh yang diutarakan penulis sebelumnya.

Token bertanda kutip dapat dimasukkan ke dalam kosakata pengenal seperti token lainnya. Jika token multi-kata bertanda

kutip tidak ditemukan dalam kosakata, maka perilaku bawaan recognizer adalah menentukan cara baca setiap token yang dipisahkan spasi di dalam tanda kutip, namun juka tidak, maka recognizer akan memperlakukan teks dalam tanda kutip sebagai token tunggal.

Untuk mencakup sebuak simbol dalam sebuah token, tanda kutip yang mengelilingi harus digunakan dan didahului oleh garis miring kiri "\". Serupa, untuk mencakup garis miring kiri dalam sebuah token bertanda kutip, harus didahului dengan garis miring kiri yang lain. Berikut ini adalah dua token yang mewakili sebuah garis miring kiri tunggal dan sebuah karakter berkutip tunggal. Harap diingat bahwa spasi dianggap penting dalam token bertanda kutip.

“\\” “\””

Kebanyakan pengenal ucapan menyajikan kemampuan menangani simbol-simbol umum dan bentuk cara pengucapan. Sebagai contoh, recognizer untuk bahasa Inggris biasanya dapat menangani tanda apostrof (Henry's, I'll) dan tanda hubung (“new-world”).

Namun, ada banyak sekali bentuk tekstual yang sulit bagi recognizer menangani kosakata tanpa ambigu. Dalam contoh berikut ini, seorang pengembang grammar harus

menggunakan token yang sedekat mungkin dengan cara kebanyakan orang akan mengucapkan dan mungkin dapat dibuat ke dalam daftar kosakata. Berikut ini contoh pada umumnya

- Angka “0 1 2 3” harus dijabarkan menjadi “nol satu dua tiga”
- Tanggal “2016-04-12” harus ditulis sebagai “tanggal dua belas bulan empat tahun dua ribu enam belas”
- Singkatan “Bpk.”, “Sdr.”, “M.T. ” masing-masing harus ditulis secara lengkap dengan bentuk “Bapak”, “Saudara”, “Magister Teknik”
- Simbol khusus seperti &, *, ^ ditulis dengan bentuk “dan”, “bintang” (atau ”kali”), “pangkat”.

4. Terakhir adalah komentar. Komentar dapat ditulis di bagian kepala maupun badan. Cara penulisannya diadopsi dari Bahasa Pemrograman Java, yaitu
   1) Dua garis miring kanan (//) yang mengabaikan sebuah baris hingga akhir
   2) Dua garis miring kanan yang diapit dengan dua tanda bintang (/**/), mengabaikan semua baris yang diapit antara /* dan */

Komentar dapat muncul di mana pun dalam penentuan grammar kecuali di dalam token, token bertanda kutip, nama aturan, dan bobot (khusus untuk bobot akan dijelaskan nanti).

Sebelum membahas tentang kaidah pembuatan grammar lebih detail, penulis menekankan secara ringkas bahwa berkas JSGF ditulis dengan dua bagian: kepala (head) dan badan (body). Bagian kepala terdiri dari tanda pengenal header (self-identifying header), deklarasi grammar (grammar declaration), dan pernyataan import. Sedangkan bagian badan terdiri dari daftar nama aturan dengan tata cara yang telah ditetapkan dalam subpoin sebelumnya.

1. Header menunjukkan bahwa dokumen mengandung JSGF dan menunjukkan versi JSGF yang sedang digunakan. (kini "V1.0"). Berikutnya, header secara pilihan merinci sistem pengkodean karakter yang digunakan dalam dokumen. Header juga dapat merinci bahasa grammar yang ditentukan dalam dokumen. Locale merinci bahasa dan negara atau daerah yang bermacam-macam yang grammar dukung. Header diakhiri dengan tanda titik koma dan karakter baru Berikut ini format penulisan header

   #JSGF version char-encoding locale;

   Berikut ini adalah contoh penulisan header seperti format yang telah diberikan.

- #JSGF V1.0;
- #JSGF V1.0 ISO8859-5;
- #JSGF V1.0 JIS ja;

Contoh pertama tidak mencakup pengkodean karakter, tidak pula locale. Sehingga, secara pengkodean karakter maupun locale akan ditetapkan secara bawaan. Di Amerika, bawaannya adalah ISO8859-1 (Sebuah himpunan karakter standar) dan "en" (simbol Inggris). Contoh kedua menetapkan kode karakter ISO8859-5, namun locale bawaan ditetapkan. Contoh ketiga menetapkan JIS (salah satu himpunan karakter dalam aksara Jepang) sebagai himpunan karakter dan menetapkan bahasa Jepang dengan simbol "ja". Karakter tanda pagar harus menjadi karakter pertama dalam dokumen dan semua karakter dalam header pengenal harus dalam sub-himpunan pengkodean ASCII.

2. Grammar. Nama Grammar harus dideklarasikan setelah header pengenal. Ada dua format yang berlaku di sini.
   - grammar packageName.simpleGrammarName;
   - grammar grammarName;

Tentang aturan penulisan grammar telah dijelaskan pada subpoin sebelumnya. Selanjutnya, berikut ini adalah contoh penulisan grammar sesuai format yang diberikan.

- grammar com.sun.speech.apps.numbers;
- grammar edu.unsw.med.people;
- grammar examples;

3. Statement import. Sifatnya pilihan dengan cara deklarasi yang mengikuti aturan deklarasi grammar. Statement ini berfungsi memanggil aturan publik dalam grammar lain agar dapat dirujuk secara lokal. Format penulisannya ada dua, yakni
   - import <fullyQualifiedRuleName>;
   - import <fullGrammarName.*>;

   Berikut ini adalah contoh penulisan dari masing-masing format
   - import <com.sun.speech.app.index.1stTo31st>;
   - import <com.sun.speech.app.numbers.*>;

   Contoh pertama adalah impor dari sebuah aturan tunggal yang ditulis lengkap.: aturan <1stTo31st> dari grammar <com.sun.speech.app.index.>. Aturan yang diimpor harus bersifat public Sedangkan pada contoh kedua, ada penggunaan tanda bintang yang menunjukkan bahwa semua aturan publik akan diimpor.

   Perlu dicatat bahwa karena baik nama grammar maupun nama aturan atau tanda bintang dibutuhkan, pernyataan impor tidak boleh ditulis dalam format berikut.

```
import <ruleName>; // salah
```

Aturan yang diimpor dapat dirujuk dengan tiga cara. Pertama, dengan nama aturan singkat <ruleName>. Kedua, dengan nama aturan sebagian <packageName.ruleName>. Ketiga, dengan menuliskan nama aturan secara lengkap <packageName.sub1.sub2....subN.ruleName>. Pernyataan impor bersifat opsional atau pilihan di saat aturan yang diimpor selalu dirujuk dengan nama aturan lengkapnya.

```
// Mengimpor com.sun.speech.app.numbers opsional
<rule> = <com.sun.speech.app.numbers.digits>;
```

Bagian kedua berkas JSGF setelah kepala (Header) adalah Badan (body). Di sinilah berbagai nama aturan dideklarasikan. Nama aturan hanya boleh ditulis satu kali, dan urutan penulisan nama aturan tidak menjadi pertimbangan dalam bagian body ini. Ada dua pola penetapan nama aturan, yakni

```
<ruleName> = ruleExpansion;
public <ruleName> = ruleExpansion;
```

Komponen dalam penetapan nama aturan adalah (1) kata kunci 'public', (2) nama aturan <ruleName> yang ditetapkan, (3) tanda sama dengan, (4) ekspansi aturan, dan (5) tanda titik koma untuk mengakhiri deklarasi aturan. Spasi kosong yang terletak sekitar kata

kunci publik, nama aturan, tanda sama dengan, dan titik koma diabaikan. Namun tetap dianggap penting di dalam ekspansi aturan.

Ekspansi aturan menetapkan cara aturan dapat diucapkan, yang merupakan kombinasi logika token (teks yang dapat diucapkan) dan acuan untuk aturan lain. Istilah ekpansi digurnakan karena sebua ekspansi menunjukkan bagaiman sebuah aturan dikembangkan saat diucapkan – sebuah aturan dapat mengembang menjadi banya kata yang diucapkan, ditambah aturan lain yang dikembangkan sendiri. Aturan penulisan ekspansi aturan akan dibahas pada bagian berikutnya.

Tiap aturan dalam sebuah grammar dapat dideklarasikan sebagai aturan publik dengan menggunakan keyword 'public'. Aturan publik memiliki tiga kegunaan yang mungkin.

- Dapat dijadikan acuan dalam penetapan aturan grammar lain dengan nama aturan lengkapnya atau dengan deklarasi impor dan sebuah bentuk acuan tanpa ambigu.
- Dapat digunakan sebagai aturan aktif untuk pengenalan (recognition). Artinya aturan dapat digunakan oleh recognizer untuk menentukan apa yang akan diucapkan.
- Dapat dirujuk secara lokal. Artinya, aturan ditetapkan dalam grammar yang sama, baik bersifat publik maupun non-publik.

Tanpa deklarasi publik, sebuah aturan bermakna privat, dan hanya bisa dirujuk dengan penetapan aturan dalam grammar lokal.

Selanjutnya adalah rincian penulisan ekspansi aturan. Ekspansi aturan sederhana adalah rujukan yang mengacu sebuah token dan rujukan yang mengacu sebuah aturan. Sebagai contoh,

- <a> = gajah;
- <b> = <x>;
- <c> = <com.hello.grammar.y>;

Aturan <a> mengembang ke arah sebuah token tunggal “gajah”. Sehingga, untuk mengucapkan <a>, pengguna harus mengucapkan kata “gajah”. Aturan <b> mengembang sebuah token <x>. Ini berarti untuk mengucapkan <b>, pengguna harus mengucapkan sesuatu yang cocok dengan aturan <x>. Sama saja ketika hendak mengucapkan aturan <c>, pengguna harus mengucapkan sesuatu yang cocok dengan aturan <com.acme.grammar.y>.

Dalam istilah lebih formal, berikut ini adalah ekspansi yang diperbolehkan.

- Token apa saja.
- Sebuah acuan kepada aturan apa saja yang ditentukan dalam grammar yang sama, baik publik maupun non-publik.
- Sebuah acuan kepada aturan publik apa saja di grammar lain yang telah diimpor ke dalam grammar terkini.

- Sebuah acuan kepada sebuah atruran publik di grammar lain saat dirujuk dengan nama aturan lengkap (dengan atau tanpa import).

Kebijakan rujukan ini ditentukan secara lokal, maksudnya dalam cakupan aturan sekarang. Contohnya, sebuah aturan dalam grammar1 secara legal dapat merujuk sebuah aturan publik di grammar2, yang selanjutnya mengacu sebuah aturan dalam grammar2 yang bersifat non-publik. Dengan kata lain, sebuah penetapan aturan dapat secara tidak langsung mengacu aturan privat pada grammar lain melalui sebuah aturan publik di grammar lain.

Di samping itu, deklarasi aturan dapat merujuk kepada token khusus (seperti NULL dan VOID), namun deklarasi aturan tidak boleh kosong.

<d> = <NULL>; // benar

<e> = <VOID>; // benar

<f> = ; // salah

Berikutnya, penulis akan menjelaskan secara detail beberapa istilah yang menunjukkan cara menjabarkan aturan menjadi lebih kompleks dengan gabungan ekspansi yang bersifat logis, yaitu

- Komposisi, terdiri dari rangkaian ekspansi (menggunakan spasi), himpunan ekspansi alternatif (menggunakan tanda garis tegak |), dan pembobotan (menyisipkan angka di antara dua garis miring kanan /#/).

- Pengelompokan, menggunakan kurung, baik biasa () maupun siku []
- Operator Unary, menunjukkan pengulangan ekspansi. Terdiri dari tanda bintang * dan tanda tambah +.
- Tag penanda sebagai lampiran aplikasi khusus.

2.2.4.1. Komposisi

2.2.4.1.1. Rangkaian Ekspansi ( )

Sebuah aturan dapat ditentukan oleh serangkaian ekspansi. Tiap ekspansi dipisah dengan spasi, dan itu merupakan ekspansi legal. Sebagai contoh, karena token maupun aturan acuan merupakan ekspansi legal, berikut ini adalah penetapan aturan yang legal.

- <tempat_tinggal> = aku tinggal di kampung;
- <pernyataan> = <objek> ini <sifat>;

Untuk mengucapkan sebuah rangkaian, tiap item dalam rangkaian harus diucapkan dalam urutan tertentu. Dalam contoh pertama, untuk menyatakan aturan <tempat_tinggal>, pembicara harus mengucapkan kata “aku tinggal di kampung” dalam urutan yang tepat. Contoh kedua mencampur token dengan merujuk aturan <object> dan <sifat>, sehingga untuk menyatakan aturan

<pernyataan>, pengguna harus mengucapkan sesuatu yang cocok dengan <objek>, diikuti dengan kata “ini”, dan akhirnya mengucapkan kata yang cocok dengan aturan <sifat>.

2.2.4.1.2. Alternatif (|)

Konsep komposisi kedua adalah alternatif. Sebuah aturan dapat ditetapkan secara alternatif dengan menuliskan serangkaian ekspansi, lalu antar-ekspansi dipisahkan dengan garis tegak (|). Spasi di sekitar garis tegak bersifat opsianal. Berikut ini contohnya.

<buah> = Apel | Jeruk | Tomat | Melon | <buahLain>;

Untuk menyatakan aturan <buah>, pengucap harus mengatakan satu dari ekspansi yang diberikan sebelumnya. Yakni Apel, Jeruk, Tomat, Melon, atau token lain yang cocok dengan aturan <buahLain> Tidak bisa mengucapkan lebih dari satu. Artinya, jika telah mengucapkan salah satu dari ekspansi yang diberikan, maka pengguna tidak dapat mengucapkan aturan yang sama secara berurutan walau pun dalam bentuk ekspansi yang sekelompok.

Serangkaian kata yang dipisahkan dengan spasi lebih didahulukan daripada bentuk alternatif. Penjelasannya dapat diambil dari salah satu contoh berikut.

<warna> = Merah Bata | Hijau Pirus | Biru Langit;

Perlu diingat bahwa tidak boleh ada alternatif yang kosong dalam menetapkan aturan. Berikut ini contoh yang salah dalam menulis ekspansi aturan alternatif.

<hewan> = kuda | | kambing; // salah

<bunga> = edelweiss | bakung | |; // salah

2.2.4.1.3. Pembobotan (weighting)

Tidak semua cara pengucapan grammar memiliki kemungkinan yang sama. Bobot dapat dilampirkan ke dalam unsur himpunan alternatif untuk menunjukkan kemungkinan tiap alternatif yang akan diucapkan. Bobot berupa nilai angka mengambang (floating point) yang dikelilingi oleh garis miring kanan, contohnya adalah /1.618/. Semakin tinggi nilainya, semakin besar kemungkinan kata yang diucapkan akan muncul. Pembobotan diletakkan sebelum tiap item dalam himpunan alternatif. Berikut ini Contoh di antaranya:

- <warna> = /4/ Merah | /2/ Kuning | /8/ Biru;

- <fase_benda> = /.2f/ Padat | /.6f/ Cair | /.3f/ Gas;
- <hewan_laut> = ikan (/25/ hiu | /10/ paus | /15/ pedang);

Bobot item harus mencerminkan kemunculan pola unsur himpunan alternatif. Dalam contoh pertama, penulis grammar sedang menunjukkan bahwa Biru 2 kali lebih mungkin muncul saat diucapkan daripada Merah dan 4 kali lebih sering muncul daripada Kuning. Syarat berikut ini harus terpenuhi saat merinci bobot.

- Jika bobot ditentukan oleh salah satu item dalam himpunan alternatif, maka bobot juga harus ditentukan untuk setiap item di dalam alternatif itu sendiri. Ini disebut sebagai prinsip semua atau tidak ada (all or nothing principle).
- Bobot ditulis dalam nilai angka mengambang, seperti 78, 1.618, 2.818e2, 8f.
- Hanya format nilai float dan spasi yang boleh disispkan di antara dua garis miring kanan.
- Bobot di bawah nol tidak diperbolehkan. Jika sama dengan nol, maka kata tersebut tidak akan diucapkan. Setara dengan aturan khusus <VOID>.

- Setidaknya satu bilangan positif bukan nol harus ditambahkan.

Dalam praktiknya, sulit menentukan bobot yang sesuai. Selain itu, menerka dan menduga bobot tidak selalu meningkatkan performa pengenalan ucapan. Bobot efektif biasanya didapatkan melalui studi pengucapan sebenarnya dan persiapan data tekstual. Di samping itu, tidak semua recognizer menggunakan bobot dalam proses pengenalan. Setidaknya, recognizer dibutuhkan untuk memastikan bahwa alternatif apa pun dengan bobot nol tidak dapat diucapkan.

2.2.4.2. Pengelompokan

2.2.4.2.1. Pengelompokan Biasa

Tiap ekspansi legal dapat dikelompokkan menggunakan tanda kurung. Pengelompokan memiliki derajat pertimbangan lebih tinggi dan dapat digunakan untuk memastikan kebenaran tafsiran aturan. Teknik ini juga berguna untuk meningkatkan kejelasan pengucapan. Sebagai contoh, karena rangkaian memiliki tingkat pertimbangan lebih tinggi daripada alternatif, tanda kurung

dibutuhkan dalam penentuan aturan berikut, sehingga token “silakan tutup” dan “silakan hapus” dibenarkan.

<perintah> = silakan (buat | edit | hapus);

Berikut ini adalah contoh yang menampilkan rangkaian tiga item, merupakan himpunan alternatif yang dikelilingi oleh tanda kurung () untuk memastikan pengelompokan yang benar.

<perintah> = (buat | edit | hapus) (satu | banyak) (program | file | folder);

Untuk mengatakan sesuatu yang sesuai dengan aturan <perintah>, pembicara harus mengatakan satu dari tiap himpunan alternatif. Sebagai contoh, mengucapkan kalimat “buat satu program”, “edit banyak folder”, atau “hapus banyak folder” memenuhi aturan <perintah>. Jika pengelompokan mengelilingi satu ekspansi tunggal, maka entitasnya ditentukan menjadi rangkaian satu item. Tidak dibenarkan ada tanda kurung yang kosong dari ekspansi:

- (mulai) //benar
- (<berhenti>) //benar
- () //salah

2.2.4.2.2. Pengelompokan Pilihan

Kurung siku dapat ditempatkan sekitar penetapan definisi apa saja untuk menunjukkan bahwa isi kurung siku bersifat opsional. Di sisi lain, tanda kurung bersifat ekuivalen untuk pengelompokan dan memiliki tingkat pertimbangan yang sama. Contohnya dapat dilihat dalam aturan berikut.

- <waktu> = [tadi | kini | nanti];
- <subjek> = (aku | kamu | dia);
- <kondisi> = (sehat | sakit | sembuh | kambuh);
- <kalimat> = <waktu> <subjek> <kondisi>;

Untuk memenuhi aturan <kalimat>, pengguna dapat menempuh dua cara. Pertama, mengucapkan salah satu kata yang terdapat dalam aturan <waktu>, lalu diikuti dengan salah satu kata yang ada di dalam aturan <subjek>, dan dilanjutkan dengan mengucapkan salah satu kata yang tertulis dalam aturan <kondisi>. Kedua, pengguna dapat langsung menuju aturan <subjek>, lalu diikuti dengan aturan <kondisi>. Ini karena aturan <waktu> bersifat pilihan, sehingga aturan <waktu> boleh diucapkan, boleh juga diabaikan.

Seperti tanda kurung biasa, tanda kurung siku tidak boleh kosong dari item, baik berupa token maupun

berbentuk aturan.

- []; //salah

2.2.4.3. Operator Unary

Dalam JSGF, operator unary dibagi menjadi 3, yaitu bintang (*), plus (+), dan tags. Secara umum, operator unary memiliki kegunaan sebagai berikut:

- Dapat disisipkan pada ekspansi aturan legal.
- Memiliki tingkat pertimbangan yang tinggi, yakni segera menlampirkan aturan ekspansi yang mendahului.
- Hanya satu operator unary yang dapat disisipkan pada tiap ekspansi aturan (ada pengecualian untuk tags).
- Spasi di antara ekspansi aturan dan operator lampiran diabaikan.

Karena tingkat pertimbangan operator unary lebih tinggi daripada rangkaian dan alternatif, tanda kurung harus digunakan untuk mengelilingi sebuah rangkaian atau himpunan alternatif untuk menyisipkan operator pada seluruh entitas.

2.2.4.3.1. Bintang (*)

Tanda bintang, bagian dari opeator unary, menandakan bahwa ekspansi boleh diabaikan, boleh pula diucapkan

baik sekali maupun berkali-kali tanpa ada batasan. Berikut ini contohnya.

<sila> = (silakan | harap | mohon)

<perintah> = <sila> * (buat | edit | hapus);

Contoh di atas menunjukkan bahwa pengguna boleh mengabaikan aturan <sila>, boleh pula mengucapkannya, baik sekali maupun berulang-ulang. Sehingga kalimat "edit", "hapus", "buat", "silakan buat", "silakan harap edit", atau "silakan harap mohon hapus" dapat dibenarkan.

Seperti kegunaan operator unary yang telah dijelaskan secara umum di muka, operator unary memiliki tingkat pertimbangan lebih tinggi daripada rangkaian dan alternatif. Dalam contoh berikut, operator berlaku pada ekspansi terdekat yang mendahului operator.

<situasi> = sedang gelap mata *;

Ekspansi yang dimaksud dalam contoh di atas adalah token "mata". Sehingga, untuk mengucapkan aturan <situasi>, pengguna boleh mengucapkan rangkaian kata "sedang gelap mata", namun hanya kata "mata" yang boleh diulang atau diabaikan. Tanda kutip dapat digunakan untuk memberikan cakupan operator *.

<instruksi> = perhatikan (kanan kiri) *;

Contoh yang diwakili aturan <instruksi> memberikan cakupan yang lebih luas terkait ekspansi yang boleh diucapkan dengan fungsi operator *. Pengguna dapat mengucapkan rangkaian “perhatikan”, ”perhatikan kanan kiri”. atau “Perhatikan kanan kiri kanan kiri”. Ketiga cara pengucapan aturan tersebut dibenarkan.

2.2.4.3.2. Tambah (+)

Operator unary kedua adalah tanda tambah (+). Operator ini menunjukkan bahwa ekspansi harus diucapkan setidaknya satu kali atau boleh berulang-ulang. Untuk lebih jelas, penulis menyajikan contoh berikut.

<perintah> = <sila> + (buat | edit | hapus);

Contoh aturan di atas mensyaratkan kata dalam aturan <sila> untuk diucapkan, sehingga membolehkan pengguna mengucapkan kalimat “silakan harap buat”, “mohon silakan hapus”, atau “harap mohon edit” dibenarkan. Namun tidak sah mengucapkan “buat”, “hapus”, atau “edit”.

2.2.4.3.3. Tags ({})

Operator unary terakhir adalah tags. Operator tag menyajikan mekanisme bagi penulis grammar untuk melampirkan informasi pada aplikasi tertentu sebagai bagian dalam penetapan aturan. Aplikasi secara khusus menggunakan tags untuk menyederhanakan pemrosesan hasil pengenalan.

Penyisipan tag tidak memengaruhi pengenalan grammar. Justru tag dilampirkan pada objek hasil yang dikembalikan oleh recognizer pada sebuah aplikasi. Antarmuka software pengenalan menetapkan mekanisme bagi pengenalan tags.

Tags merupakan untai yang dibatasi dengan tanda kurung kurawal {}. Semua karakter yang berada dalam tanda kurung kurawal dianggap sebagai bagian dari tag, termasuk spasi. Berbeda dengan tanda kurung biasa maupun siku, tag boleh kosong dari item. Ini berarti tag akan ditafsirkan sebagai string dengan panjang nol, setara dengan “” dalam bahasa Java.

Untuk memuat tanda kurung penutup dalam tanda kutip, tanda kurung penutup harus didahului garis miring kiri \. Hal yang sama berlaku untuk garis miring kiri itu sendiri. Berikut ini contohnya

{{salam \\ halo \\ mari \}};

Contoh di atas akan diproses dengan rangkaian string berikut

"{salam \ halo \ mari }";

Tag menyertai ekspans langsung terdahulu (spasi yang menyela di antara string diabaikan). Contoh penulisan tag dapat dilihat berikut ini

- <aturan> = aksi {tag di dalam sini};
- <sila> = mohon (buka {BUKA} | tutup {TUTUP}) pintu;
- <keluarga> = bapak {Bpk.}| saudara {Sdr.} | saudari {Sdr.} | nyonya {Ny.};
- <gelar> = doktor {Dr.} | profesor {Prof.};

Sebagai operator unary, tag memiliki tingkat pertimbangan lebih tinggi daripada rangkaian dan alternatif. Berikut ini contoh yang akan diberikan penulis

1. <objek> = kursi | meja | lampu {benda};
2. <objek> = (kursi | meja | lampu) {benda};

Pada contoh pertama, aturan <benda> mengandung tag yang hanya berlaku untuk item "lampu". Sedangkan tag pada contoh kedua berlaku untuk serangkaian item yang terdapat di antara tanda kurung. Berbeda dengan opearor

unary lainnya, banyak tag boleh menyertai ekspansi aturan. Bahkan, tag bersarang dalam kurung juga dibenarkan. Namun, tag tidak boleh digabung dalam serangkaian operator unary yang tidak sejenis. Berikut ini contoh implementasi tag

- <terjun> = jatuh {tersungkur} {terjungkal} {terjungkir}; //benar
- <zat> = ((foton {partikel}) {sub-atom}) {atom}; //benar
- <aturan> = <tindakan> * {tag1} ; //salah
- <aturan> = <tindakan> {tag1} +; //salah

Fungsi tag adalah menyederhanakan aplikasi dengan meringkas proses hasil pengenalan. Isi tags dan penggunaan tags bergantung kepentingan pengembang. Salah satu penggunaan penting sebuah tag adalah aplikasi yang bersifat internasional. Berikut ini adalah contoh penetapan aturan untuk empat grammar, tiap grammar untuk bahasa yang terpisah. Aplikasi memuat grammar untuk bahasa yang digunakan oleh pengguna ke dalam recognizer yang sesuai. Isi tag tetap sama pada seluruh bahasa dan kemudian menyederhanakan software aplikasi yang memproses hasil pengenalan. Secara khusus, nama

grammar akan mencakup pengenal bahasa (identifier language) sehingga dapat ditemukan dan dimuat secara program.

- <greeting> = (howdy | good morning) {hi}; //Inggris
- <greeting> = (ohayo | ohayogozaimasu) {hi}; //Jepang
- <greeting> = (bonjour) {hi}; //Perancis
- <greeting> = (guten tag) {hi}; //Jerman

Dari teknik penjabaran aturan yang telah disinggung di muka, penulis akan mengurutkannya berdasarkan tingkat pertimbangan ekspansi dalam JSGF, dari tinggi ke rendah

1. Nama aturan (dikelilingi tanda kurung sudut <>), dan token, baik bertanda kutip maupun tanpa tanda kutip.
2. Pengelompokan (baik mandatoris '()' maupun pilihan '[]').
3. Operator unary('+', '*', '{tag}') yang berlaku pada ekspansi aturan langsung terdahulu yang paling dekat dengan operator.
4. Rangkaian ekspansi aturan yang dipisahkan dengan spasi.
5. Himpunan ekspansi alternatif yang dipisahkan dengan garis tegak '|'.

2.2.4.4. Teknik Lanjutan

Konsep manajemen ekspansi lebih lanjut adalah rekursi dan nama aturan khusus. Rekursi adalah penetapan aturan dalam dirinya sendiri. Sedangkan nama aturan khusus telah dibahas di muka, yakni <NULL> dan <VOID>. Penulis akan menjabarkan aturan khusus lebih lanjut setelah konsep rekursi.

2.2.4.4.1. Rekursi

Rekursi merupakan teknik yang memungkinkan perwakilan banyak bentuk grammatikal kompleks yang muncul dalam bahasa yang digunakan. Recognizer yang mendukung JSGF hanya memperbolehkan rekursi sebelah kanan. Dalam rekursi sebelah kanan, aturan mengacu pada dirinya sendiri sebagai bagian terakhir penetapan ekspansi aturan.

- <perintah> = <tindakan> | (<tindakan> terus <perintah>);
- <tindakan> = (berhenti | maju | mundur | kanan | kiri);

Dalam contoh di atas, di antara perintah berikut yang boleh diucapkan adalah berikut ini:"mundur lalu kanan lalu berhenti", "kiri lalu maju lalu kanan lalu mundur", "maju terus kanan terus maju terus kiri terus berhenti" .

Rekursi sebelah kanan dapat dibuat secara bersarang. Maksudnya adalah antar aturan yang saling merujuk satu sama lainnya, dengan tiap acuan rekursif menjadi bagian akhir penetapan aturan. Rekursi semacam ini dapat tampil di berbagai grammar. Kendati demikian, rekursi bersarang ini sangat tidak dianjurkan, karena dapat memperumit pengenalan dan menyebabkan masalah potensial dalam pengenalan kata. Berikut ini contoh rekursi bersarang

- <X> = suatu hal | <Y>;
- <Y> = lain hal <X>;

Aturan rekursif kanan apa pun dapat ditulis ulang menggunakan operator bintang * dan tambah +. Meski hal tersebut dimungkinkan, bentuk rekursif diizinkan karena bentuk penulisan tersebut dapat memperkenankan representasi beberapa grammar yang lebih sederhana dan lebih elegan.

- <perintah> = <tindakan> | (<tindakan> terus <perintah>);
- <perintah> = <tindakan> (terus <tindakan>) *;

Bentuk rekursif lainnya adalah rekursi kiri dan rekursi tertanam, namun JSGF tidak mendukung rekursi ini karena syarat penulisan ulang tidak dapat dijamin kebenarannya.

Secara teknis JSGF menetapkan bentuk grammar bernama tata bahasa beraturan. Beberapa fitur yang dikaitkan dengan tata bahasa bebas konteks diizinkan untuk kejelasan dan kemudahan.

2.2.4.4.2. Aturan Khusus (<NULL> dan <VOID>)

Teknik lanjutan kedua adalah aturan khusus. Sebuah acuan <NULL> sebagai alternatif bagi ekspansi aturan apa pun sama dengan penggunaan tanda kurung siku sebagai pengelompokan pilihan. Acuan <NULL> dapat pula dipetakan secara alternatif dengan konsep rekursif, sehingga setara dengan penggunaan tanda bintang sebagai opeator unary pada ekspansi aturan yang bersangkutan. Penjelasan tersebut dapat diwakilkan dengan contoh berikut

1. <x> = a | <NULL>;

   <x> = [a];
2. <x> = <NULL> | a <x>;

   <x> = a*;

Untuk kedua kasus sebelumnya, grammar di atas bermakna sama, dengan alasan pengguna dapat berkata tepat dengan penyebutan yang sama. Meski demikian, akan

ada perbedaan terprogram dalam gambaran hasil yang diproduksi oleh recognizer saat pengguna mengatakan sesuatu yang cocok dengan grammar.

Fungsi terakhir kedua aturan khusus adalah pemagaran (*gating*). Untuk menyalakan dan mematikan bagian grammar tertentu, aplikasi dapat menentukan penggunaan aturan <pagar>, letakkan sebelum bagian grammar yang akan dinyalakan atau dimatikan, dan ganti definisi <pagar> antara:

- <pagar> = <NULL>;
- <pagar> = <VOID>;

Saat <pagar> sama dengan <NULL>, ekspansi dalam grammar yang bersangkutan akan menetapkan aturan yang cocok secara otomatis, tanpa harus mengucapkan kata apa pun oleh pengguna. Namun, jika <pagar> sama dengan <VOID>, maka rangkaian grammar tidak dapat diucapkan dan bagian grammar yang menggunakan aturan <pagar> akan dimatikan.

Sebagai bentuk kehati-hatian, teknik ini membutuhkan recognizer yang dapat menyusun ulang grammar secara efisien setelah perubahan definisi <pagar> agar dapat

bekerja efektif. Di samping itu, tidak semua recognizer dapat mengerjakannya secara efisien.

2.2.5. Hadits

2.2.5.1. Definisi Hadits

Dalam bahasa Arab (khususnya makna literal / harfiah), kata hadits (حديث) mengandung banyak makna, di antaranya, semakna dengan kata jadid (جديد) yang berarti 'baru' dan lawan kata qadim (قديم) yang bermakna 'lama' / 'dulu'. Sinonim lain kata ini adalah qaul (قول), yang secara literal mengandung makna 'ucapan'. Menurut istilah, kata hadits mengalami penyempitan makna menjadi perkataan, perbuatan, atau persetujuan yang disandarkan kepada Rasulullah. Dengan demikian, terminologi (istilah) tersebut bermakna sama dengan sunnah (سنة), namun perbedaannya terletak pada definisi harfiahnya. Kata sunnah (سنة) mengandung makna jalan atau cara. Bila pengertian hadits hanya dipahami secara lebih sempit sebagai 'ucapan' atau 'perkataan', maka sunnah lebih ditekankan kepada perbuatan, sehingga kedua istilah tersebut bersifat saling melengkapi. Terkait pengertian terminologi sunnah, ada tiga perbedaan terkait hal tersebut. Hal ini karena

definisi tersebut dipengaruhi oleh disiplin ilmu yang menggunakan istilah tersebut, antara lain

1. Ilmu Hadits (علوم الحديث): Kata sunnah memiliki kesamaan arti dengan Hadits. Secara istilah, kata ini bermakna seperti yang telah dikemukaan di muka, namun terminologi ini berlaku secara luas, baik sebelum maupun sesudah periode kenabian
2. Ilmu Ushul Fiqh (اصـل الفقـه): Kata sunnah juga serupa dengan definisi dalam ilmu Hadits, namun ketentuan tentangnya hanya terbatas pada masalah perumusan dan penerapan hukum. Dengan demikian, istilah sunnah dalam disiplin ilmu Ushul fiqh menjadikan sunnah sebagai sumber hukum setelah al-Qur'an.
3. Ilmu Fiqh (علوم الفقه): Kata sunnah mengandung dua makna sekaligus. Pertama seperti penjelasan definisi Hadits. Kedua, kata sunnah termasuk bagian hukum itu sendiri, di samping wajib, mubah, makruh, dan haram. Dari pengertian yang kedua pada disiplin ilmu fiqh, kata ini merupakan sinonim kata *mustahabb* (مستحب), yang berarti 'dicintai'. Dimaknai demikian karena segala hal yang dianggap sunnah adalah hal yang dicintai oleh Rasul. Dengan demikian, definisi istilah sunnah dalam perngertian kedua

adalah perbuatan yang mendatangkan pahala bila dikerjakan, dan tidak berdosa bila ditinggalkan.

Jika di muka dijelaskan bahwa hadits mencakup perkataan, perbuatan, dan persetujuan yang disandarkan kepada Nabi, sedangkan secara harfiah lawan kata hadits (baru) adalah qadim yang bermaknal lama, maka kali ini pengertian lawan kata sunnah dapat dikaitan dengan pengertian hadits secara bahasa, yaitu segala macam ucapan, perbuatan, dan persetujuan yang baru dan sama sekali tidak bersumber dari Rasul. Para ahli hadits mendefinisikan lawan kata yang dimaksud sebagai bid'ah (بدعة). Tentu saja, definisi bid'ah dalam tinjauan ilmu hadits hanya mencakup masalah agama, bukan termasuk masalah duniawi. Karena terkait masalah dunia, Rasul menegaskan bahwa boleh jadi umatnya lebih memahami dari Rasul.

أنتم تعلمون بأمور دنياكم

"Kalian lebih mengetahui urusan dunia kalian"

Hadits di atas biasa dipakai sebagai dalil bahwa urusan dunia diserahkan kepada masing-masing individu yang ahli di bidang yang bersangkutan. Namun, itu bukan berarti tidak memerlukan bimbingan dari Rasul, karena bagaimana pun manusia memiliki keahlian dalam berbagai hal, Rasul lebih

mengetahui maslahat dan hikmah yang ditimbulkan dari masing-masing urusan. Oleh karena itu pula, hadits di atas tidak dapat dijadikan alasan menghindari kewajiban dalam mencari tahu maslahat dan hikmah yang terkandung di dalamnya. Pendek kata, hadits di atas tidak dapat dijadikan dasar untuk melepaskan diri dari kewjiban beragama secara umum. Terkait dasar penetapan hadits sebagai sumber hukum, penulis akan menjelaskannya pada subbab berikutnya. Di samping itu, ada sebuah hadits yang menerangkan bahwa Rasul lebih peduli terhadap umatnya daripada umat itu sendiri.

إنّما مثلي و مثل الناس كمثل رجل استوقد نارًا فلمّا أضاءتْ ما حوله جعل الفراش و هذه الدواب التي تقع فى النار يقعن فيها فجعل ينزعهُنَّ و يغلبنه فيقتحمن فيها فأنا آخذ بحجزكم عن النار و هم يقتحمون فيها

"Perumpamaan diriku dengan orang lain seperti seorang yang menyalakan api. Ketika segala hal yang di sekitarnya terang, maka serangga dan kupu-kupu jatuh ke dalam api. Kemudian orang itu berupaya menghalau kupu-kupu dan lalat agar tidak terjerumus ke dalam api tersebut. Namun mereka dapat mengalahkan orang itu sehingga terjerumus ke dalam api. Sebagaimana aku menarik ikat pinggang kalian agar tidak masuk neraka, namun tetap saja kalian menerjangnya." (Muttafaq Alaih)

2.2.5.2. Alasan Penetapan Hadits sebagai Sumber Utama Hukum Islam

Meyakini ajaran Islam melalui al-Qur'an merupakan bukti nyata keimanan seseorang bahwa tiada Tuhan selain Allah. Sedangkan meyakini ajaran Islam melalui Hadits merupakan bukti nyata keimanan seseorang bahwa Muhammad adalah Rasul Allah. Itulah manifestasi kalimat tauhid لا إله إلا الله محمد رسول الله (Tiada Tuhan selain Allah, Muhammad Rasul Allah) secara menyeluruh. Oleh karena itu, penulis memaparkan berbagai alasan beserta dalil terkait keharusan kaum muslimin menjadikan hadits sebagai rujukan utama setelah al-Qur'an

1. Allah mewajibkan kaum muslimin menaati Rasulullah.

يآ اَيُّهَا الَّذِيْنَ آمَنُوْا اَطِيْعُوا اللهَ وَ اَطِيْعُوْا الرَّسُوْلَ وَ اُولِى الْأَمْرِ مِنْكُمْ . فَإِنْ تَنَازَعْتُمْ فِيْ شَيْءٍ فَرُدُّوْهُ اِلَى اللهِ وَ اِلَى الرَّسُوْلِ اِنْ كُنْتُمْ تُؤْمِنُوْنَ بِا اللهِ وَ الْيَوْمِ الْآخِرِ . ذَلِكَ خَيْرٌ وَ اَحْسَنُ تَأْوِيْلًا (النساء: 59)

"*Wahai orang-orang beriman, patuhilah Allah, patuhilah Rasul Allah dan para pemegang kekuasaan di antara kalian. Maka jika kalian berbeda pendapat tentang sesuatu, maka kembalikan permasalahan tersebut kepada Allah dan Rasulullah jika kalian*

beriman kepada Allah dan hari akhir. Itu lebih baik dan merupakan penakwilan yang terbaik." (QS. An-Nisa' [4] : 59)

2. Allah menjadikan kepatuhan terhadap Rasulullah sama dengan kepatuhan kepada Allah

مَنْ يُطعِ الرَّسُوْلَ فَقَدْ اَطَاعَ اللهَ . وَمَنْ يَتَوَلَّى فَمَا اَرْسَلْنَاكَ عَلَيْهِمْ حَفِيظًا (النساء: 80)

"Siapa yang mematuhi Rasul, maka sungguh ia telah mematuhi Allah. Dan siapa yang berpaling, maka kami tidak akan mengutusmu (Muhammad) sebagai pemelihara bagi mereka." (QS. An-Nisa' [4] : 80)

3. Allah menjadikan kepatuhan terhadap Rasulullah sebagai prasyarat cinta terhadap Allah

قُلْ اِنْ كُنْتُمْ تُحِبُّوْنَ اللهَ فَاتَّبِعُوْنِيْ يُحْبِبْكُمُ اللهُ وَ يَغْفِرْ لَكُمْ ذُنُوْبَكُمْ . وَ اللهُ غَفُورٌ رَّحِيْمٌ (آل عمران: 31)

"Katakanlah, jika kalian mencintai Allah, maka ikutilah aku (Nabi Muhammad), niscaya Allah mencintai kalian dan mengampuni dosa-dosa kalian. Dan Allah Maha Pengampun, Maha Penyayang." (QS. Ali Imron [3] : 31)

4. Rasulullah merupakan teladan terbaik bagi seluruh manusia

لَقَدْ كَانَ لَكُمْ فِيْ رَسُوْلِ اللهِ اُسْوَةٌ حَسَنَةٌ لِمَنْ كَانَ يَرْجُوْا اللهَ وَ الْيَوْمَ الآخِرَ وَ ذَكَرَ اللهَ كَثِيْرًا (الاحزاب: 21)

"Sungguh, ada keteladanan yang baik di dalam (diri) Rasulullah untuk kalian, yakni bagi orang-orang yang mengharapkan (rahmat) Allah dan (kedatangan) hari Akhir, serta bagi orang yang banyak mengingat Allah." (QS. Al-Ahzaab [33] : 21)

5. Allah memuji Rasulullah sebagai sosok yang memiliki akhlak mulia

وَ اِنَّكَ لَعَلَيْ خُلُقٍ عَظِيْمٍ (القلم: 4)

"Dan sungguh, kamu (wahai Muhammad) benar-benar berbudi pekerti yang luhur (QS. Al-Qolam [68] : 4)"

6. Tidak idealnya keimanan tanpa menjadikan Rasulullah sebagai pemutus perkara terbaik untuk seluruh manusia

فَلاَ وَ رَبِّكَ لاَ يُؤْمِنُوْنَ حَتّى يُحَكِّمُوْكَ فِيْمَا شَجَرَ بَيْنَهُمْ ثُمَّ لاَ يَجِدُوْا فِيْ اَنْفُسِهِمْ فِيْمَا قَضَيْتَ وَ يُسَلِّمُوْا تَسْلِيْمًا (النساء: 65)

"Maka, demi Tuhanmu, mereka tidak beriman sebelum mereka menjadikanmu hakim terhadap perkara yang mereka perselisihkan, lalu mereka tidak mendapati adanya keberatan dalam diri mereka terkait ketetapan

yang kau berikan, sehingga mereka menerima sepenuhnya." (QS an-Nisaa [4] : 65)

7. Perintah Allah kepada kaum muslimin adalah mengikuti ajaran yang dibawa Rasul.

مَآ اَفَآءَ اللّٰهُ عَلٰى رَسُوْلِهٖ مِنْ اَهْلِ الْقُرٰى فَلِلّٰهِ وَ لِلرَّسُوْلِ وَ لِذِى الْقُرْبٰى وَ الْيَتٰمٰى وَ الْمَسٰكِيْنِ وَ ابْنِ السَّبِيْلِ كَيْلَا يَكُوْنَ دُوْلَةً بَيْنَ الْاَغْنِيَآءِ مِنْكُمْ . وَ مَآ اٰتٰىكُمُ الرَّسُوْلُ فَخُذُوْهُ وَ مَا نَهٰىكُمْ عَنْهُ فَانْتَهُوْا . وَ اتَّقُوا اللّٰهَ . اِنَّ اللّٰهَ شَدِيْدُ الْعِقَابِ (الحشر:7)

"Harta rampasan dari mereka yang telah Allah berikan kepada Rasul-Nya (yang berasal) dari penduduk beberapa negeri, adalah untuk Allah, Rasul-Nya, kerabat dekat, anak-anak yatim, orang-orang miskin, dan untuk orang-orang yang dalam perjalanan, agar (harta tersebut) tidak beredar pada orang kaya di antara kalian saja. Terimalah apa yang Rasul berikan kepada kalian, dan tinggalkanlah apa saja yang Rasulullah larang untuk kalian. Dan bertakwalah kepada Allah, sesungguhnya siksa Allah sangat keras." (QS. Al-Hasyr [59] : 7)

8. Rasulullah adalah penafsir al-Qur'an terbaik.

وَ اَنْزَلْنَا اِلَيْكَ الذِّكْرَ لِتُبَيِّنَ لِلنَّاسِ مَا نُزِّلَ اِلَيْهِمْ لَعَلَّهُمْ يَتَفَكَّرُوْنَ

(النحل: 44)

"Dan kami telah menurunkan adz-Dzikr (al-Qur'an) agar kamu menjelaskan kepada manusia apa yang diturunkan kepada mereka agar mereka berpikir." (QS. An-Nahl [16] : 44)

2.2.5.3. Hadits-hadits pilihan

Berikut ini merupakan lima contoh hadits yang akan diujikan melalui proposal ini

1. Hadits tentang keutamaan sifat malu

الحياء لا يأتي إلا بخير

Sifat malu hanya akan mendatangkan kebaikan

2. Hadits tentang amalan Islam yang paling utama

من سلم المسلمون من لسانه و يده

Orang muslim yang selamat baik lisan maupun tangannnya (perbuatannya)

3. Hadits tentang keimanan yang sempurna

أكمل المؤمنين إيمانا أحسنهم خلقا

Kesempurnaan iman kaum mukmin terletak pada akhlak terbaik mereka

4. Hadits tentang sifat indah Allah

ان الله جميل يحب الجمال

Sesungguhnya Allah maha Indah, Dia menyukai keindahan

5. Hadits tentang keutamaan orang berderma

اليد العليا خير من يد السفلى

Tangan di atas lebih baik daripada tangan di bawah.

BAB III

PERANCANGAN SISTEM

3.1. Desain Penelitian

3.1.1. Objek Penelitian

Penulis memaparkan objek penelitian dalam proposal ini berupa suara dari mikrofon, kemudian suara yang telah terinputkan akan diekstrak 'di balik layar' (*backend*) yang artinya dilakukan secara tersembunyi oleh framework yang akan dipakai dalam pengenalan ucapan, Suara tersebut diterjemahkan menjadi kode biner (*encoding*), sehingga dapat dimengerti oleh framework dan selanjutnya diterjemahkan kembali dalam bentuk teks (*decoding*) yang akan menjadi output terakhir, dan inilah yang akan ditampilkan oleh program sebagai hasil pengolahan suara.

Penulis menegaskan bahwa penelitian ini tidak berkaitan langsung dengan analisis komputasi yang terjadi pada saat proses pengenalan ucapan, karena seluruh proses perhitungan dilakukan oleh framework. Oleh karena itu, hasil komputasi yang dilakukan oleh framework bukan prioritas utama dalam penelitian ini. Di sisi lain, penulis hanya lebih menitikberatkan pada aspek implementasi dalam penelitian ini.

3.1.2. Variabel Penelitian

Pada subbab 3.1.1, penulis menguraikan deskripsi singkat mengenai pemrosesan objek penelitian berupa suara dan pemrosesan berakhir dengan bentuk teks. Secara ringkas, penulis menentukan beberapa variabel yang akan diguanakan untuk menunjang penelitian ini, yaitu

1. Variabel Bebas
   - Suara, direkam langsung dari mikrofon laptop
2. Variabel Terikat
   - Teks. Variabel ini adalah output yang merupakan hasil pemrosesan data suara sesuai dengan variabel kontrol yang diberikan.

3.1.3. Sumber Data

Jika di berbagai penelitian tentang pengolahan suara melibatkan analisis perhitungan metode mulai dari ekstraksi fitur hingga pengenalan dan pencocokan pola, penulis hanya terbatas pada ranah implementasi sebagaimana ditekankan dalam subbab 3.1.1. Oleh karena itu, sumber data yang diperoleh hanya ada tiga.

1. Suara yang diucapkan secara langsung dari pembicara
2. Berkas (*file*) berupa kamus bahasa yang mencakup cara pembacaan (*pronounciation*)

3. File berisi aturan tata bahasa yang dipakai sebagai acuan kata yang keluar sebagai output.

Kedua file terakhir dibuat secara manual berdasarkan aturan yang dipakai dalam pustaka Sphinx, sehingga program tersebut dapat mengenali kata yang dipakai dalam program. Sedangkan berkas suara acuan direkam melalui komputer dengan frekuensi sampel 16 kHz, bit rata-rata 16 bit, dan bertipe mono. Ketiga file di atas (suara, dictionary, dan language model [baik berupa berkas berekstensi .lm maupun berkas berekstensi .jsgf) akan dipakai untuk pemodelan akustik. Semua data yang telah disebutkan di atas bersifat primer karena ketiganya merupakan syarat mutlak yang harus dipenuhi dalam penelitian ini.

3.1.4. Instrumen Penelitian

Tidak seperti penelitian pengenalan ucapan pada umumnya yang memanfaatkan instrumen lebih kompleks karena disertai dengan analisis perhitungan suatu metode, penulis membatasinya hanya di bidang implementasi, sehingga proses pengenalan ucapan dalam aplikasi yang penulis tawarkan dapat diterapkan. Dalam penelitian ini, penulis memanfaatkan instrumen penelitian dari segi perangkat, yang dibagi menjadi 2, yaitu

1. Perangkat Keras. Penulis hanya menggunakan satu perangkat keras berupa laptop dengan spesifikasi berikut
    - CPU: AMD A6-3400 Quad Core 1.6 Ghz
    - GPU: AMD Radeon HD 6720 1 GB
    - Memori: Samsung DDR3 8 GB
    - HDD: WD 750 GB
2. Perangkat Lunak. Penulis menggunakan berbagai perangkat lunak yang menunjang penelitian ini, yaitu
    1. Netbeans 8.0, sebagai sarana pemrograman bahasa Java. Java Development Kit (JDK) dibutuhkan untuk pengembangan aplikasi bahasa Java, sedangkan Java Runtime Environment (JRE) dibutuhkan untuk menyediakan lingkungan tempat kode-kode berbahasa Java dieksekusi. Kedua framework tersebut harus sudah terinstal sebelum Netbeans.
    2. CMUCLMTK (Carnegie Mellon University Cambridge Language Model Toolkit), framework yang berguna dalam pembuatan Language Model. Perintah yang terdapat dalam framework ini dieksekusi melalui terminal ubuntu (atau command prompt dalam windows). Prosedur instalasi dapat dilihat dalam bagian

dokumentasi yang tertera dalam folder yang bersangkutan (README.txt).

3. Sphinxbase. Framework ini akan dipakai untuk membuat database yang merupakan bagian dari prosedur pelatihan model akustik. Framework ini juga dapat dipakai untuk melakukan perubahan data (konversi) dari bentuk teks (.lm) menjadi bentuk binary (.bin). Perintah yang terdapat dalam framework ini dieksekusi melalui terminal ubuntu (atau command prompt dalam windows). Prosedur instalasi dapat dilihat dalam bagian dokumentasi yang tertera dalam folder yang bersangkutan (README.txt).
4. Sphinxtrain. Framework ini mensyaratkan adanya sphinxbase yang telah terinstal sebelumnya sebagai data acuan dalam proses pembuatan model akustik selanjutnya. Di samping itu, framework ini sangat berguna dalam mengeksekusi proses pelatihan dan pengujan model akustik. Perintah yang terdapat dalam framework ini dieksekusi melalui terminal ubuntu (atau command prompt dalam windows) Prosedur instalasi dapat dilihat dalam bagian dokumentasi yang tertera dalam folder yang bersangkutan (README.txt).

5. File berekstensi .jar yang mendukung perancangan sistem pengenalan ucapan dalam pemrograman dengan bahasa java. File yang dimaksud adalah
    1. sphinx4-core-5prealpha.jar. File ini mengandung class/interface beserta isinya yang akan dipanggil dalam kode yang dibuat penulis.
    2. sphinx4-core-5prealpha-javadoc.jar. File ini berisi dokumentasi yang berguna untuk menerangkan deskripsi kelas/interface dalam file sphinx4-core-5prealpha.jar.
    3. sphinx4-core-5prealpha-sources.jar, berisi contoh source code yang dipakai sebagai demonstrasi bagaimana sistem pengenalan ucapan dapat berjalan.

    Ketiga berkas jar tersebut ditambahkan ke dalam project Netbeans agar memanggil class/interface java yang dibutuhkan dalam proses pengenalan ucapan.

    Sistem operasi yang dipakai adalah Zorin OS 9 (turunan dari Ubuntu 14.04 LTS).

3.1.5. Metode Penelitian

Penulis menggunakan dua metode penelitian dalam penulisan skripsi ini, yakni kuantitatif dan eksperimental (Sugiyono, 2008). Disebut kuantitatif karena cakupan data yang dipakai dalam proses pengenalan ucapan ini dapat ditetapkan dengan jelas dan pasti (baik dari segi jumlah data yang dipakai, maupun dari segi perlakuan yang diberikan penulis), tanpa ada upaya interpretasi yang bertujuan mencari makna di balik data. Dalam penelitian penulis, data yang dipakai pada tahap pengujian adalah suara pengucapan hadits yang telah disebutkan dalam bab 2. Sedangkan alasan penggunaan metode eksperimental adalah penelitian ini akan memanipulasi berbagai variabel yang akan diujicoba dalam sistem. Variabel yang dimaksud telah disebutkan dalam subbab 3.1.2.

3.2. Prosedur Penelitian

Jika di akhir subbab 3.1 penulis telah menjelaskan dua metode yang memaparkan deskripsi umum tentang penelitian yang akan dilakukan, maka dalam subbab 3.2 ini penulis menjabarkan langkah yang ditempuh dalam penelitian ini. Oleh karena itu, agar prosedur penelitian ini berjalan dengan sistematis, penulis menguraikan berbagai langkah yang akan dilalui sebagai berikut.

3.2.1. Pemahaman Sistem dan Studi Literatur

Tahap ini merupakan awal proses penelitian, dimulai dari menelusuri cara kerja metode yang dipakai dalam penelitian ini, kemudian menelaah berbagai parameter pada sistem yang hendak dibangun, ditunjang dengan berbagai jurnal dan kajian teoritis tentang metode terkait. Selain itu, mempelajari konsep dan model matematika merupakan hal yang tidak kalah urgen, mengingat metode yang akan dipakai dalam penelitian ini tidak lepas dari berbagai aksioma dan teorema matematika. Penulis mencoba membatasi lingkup penelitian dalam proposal ini hanya pada hal-hal yang bersifat aplikatif, sehingga cara kerja sistem diinginkan dapat dipahami dan hal-hal matematis yang berada di luar keahlian sebagian pembaca dapat segera diminimalisir. Dengan demikian, alur kerja pengenalan ucapan dengan menggunakan framework Sphinx tidak membingungkan pembaca yang masih awam terhadap konsep dan model matematika.

3.2.2. Pengumpulan Data

Data yang akan disiapkan penulis akan dikelompokkan berdasarkan komponen yang akan dibutuhkan dalam proses pengenalan ucapan, yakni Language model, Dictionary dan Acoustic Model. Dalam penelitian ini, hanya dictionary dan JSGF

yang akan dibuat secara manual, sedangkan Language Model dan Acoustic Model akan dirancang dengan tools bantu, yang akan dibahas dalam desain sistem.

1. Language Model

   Data yang dipersiapkan dalam menyusun language model ada dua

   - File text dengan satu kalimat untuk setiap baris diawali dengan tag pembuka kalimat<s> dan diakhir dengan tag penutup kalimat </s>, sehingga format isi file text yang akan diolah adalah

     <s> [kalimat apa saja] </s>

   - File .jsgf ysng dibuat secara manual dengan kaidah yang telah dibahas secara detail pada bab 2

2. Dictionary Model

   Data yang dipersiapkan dalam menyusun dictionary hanya satu, yakni daftar kata (vocabulary), diikuti oleh kumpulan fonem yang dipisahkan dengan spasi untuk setiap baris, sehingga format isi dictionary adalah

   [kata]<tab>[fonem-1]<spasi>[fonem-2]<spasi>...<spasi>[fonem-n]

   Sekali lagi, data dictionary dibuat secara manual, dengan membubuhkan fonem yang terdapat dalam file berisi daftar

fonem untuk masing-masing baris. Fonem yang tertera dalam file dictionary harus terdaftar dalam file .phone. Berkas phone akan dijelaskan lebih rinci dalam persiapan pembuatan model akustik.

3. Acoustic Model

   Data yang dipersiapkan untuk membuat model akustik adalah

   1. Output dua data yang telah disebutkan di muka (Language Model dan Dictionary). Untuk Language Model berekstensi lm, sebelumnya harus diubah lebih dahulu menjadi DMP Format dengan perintah yang telah disebutkan dalam proses pembuatan language model.
   2. Data teks berupa daftar fonem berekstensi .phone. Fonem yang tertera harus terdapat dalam dictionary dan file transcription demi terpenuhinya persyaratan untuk pembuatan model akustik, tidak boleh lebih, juga tidak boleh kurang
   3. Data teks berupa daftar kata yang berada di l;uar cakupan dictionary, dengan ekstensi .filler. Filler yang terdapat dalam dictionary juga harus memenuhi persyaratan dalam file berekstensi .phone

4. Data teks berupa transkripsi file, tersusun atas kalimat yang berbentuk seperti isi berkas input dalam membuat language model, dipisahkan dengan spasi, dan diakhiri dengan file suara acuan tanpa ekstensi. Berkas dengan ekstensi .transcription ini akan digunakan untuk kepentingan training dan testing. Untuk membedakannya, nama file ini disisipkan dengan kata _train dan _test sebelum ekstensi file yang bersangkutan. Dengan demikian, format yang berlaku dalam isi berkas transcription adalah sebagai berikut

   <s> [kalimat_apa_saja] </s> (file_suara_[n])

   Tanda [n] mewakili angka berapa pun dan harus dibuat berurutan. Di samping itu, kalimat yang diapit tanda <s> dan </s> harus sesuai dengan suara yang terdapat dalam file suara yang digunakan untuk transcription.
5. Data teks berisi file id yang berisi folder, diikuti dengan garis miring / dan diakhiri dengan nama file suara tanpa eksensi. File yang berekstensi fileids ini digunakan untuk kepentingan yang sama dengan data .transcription. Format yang berlaku dalam isi berkas fileids adalah

   speaker_[n]/file_suara_[n]

Tanda [n] mewakili angka berapa pun dan harus dibuat berurutan

6. Data suara berekstensi .wav seperti spesifikasi yang dijelaskan di muka, terkumpul dalam satu kelompok folder terpisah dari data yang telah disebutkan sebelumnya. Data suara ini akan dipakai sebagai acuan yang terdapat dalam folder transcription dan fileids, dengan ketetapan frekuensi sampling 16000 Hz, Bit Rate 16 bit, dengan tipe mono. Durasi suara yang diperbolehkan antara 5 detik hingga 30 detik untuk setiap kalimat dan jeda waktu total 0,2 detik, baik di awal, di tengah, maupun di akhir. Ini adalah standar untuk pemodelan akustik berbasis desktop. Dalam kondisi ideal, durasi total untuk semua berkas suara minimal adalah 1 jam untuk satu orang pembicara dengan pelatihan selama satu bulan.

Dengan demikian, susunan folder beserta berkas yang disiapkan untuk pembuatan model akustik adalah sebagai berikut

- etc
  - *nama_file*.dic
  - *nama_file*.filler
  - *nama_file*.jsgf
  - *nama_file.lm*.DMP

- *nama_file*.phone
- *nama_file_test*.fileids
- *nama_file_test*.transcription
- *nama_file_train*.fileids
- *nama_file_train*.transcription

- wav
  - speaker_1
    - *file_suara*_0001.wav
    - *file_suara*_0002.wav
    - *file_suara*_0003.wav
    - .....
    - *file_suara*_000n.wav
  - speaker_2
    - *file_suara*_0001.wav
    - *file_suara*_0002.wav
    - *file_suara*_0003.wav
    - .....
    - *file_suara*_000n.wav

Sekali lagi, penulis menekankan bahwa semua data yang tertera dalam subbab ini bersifat primer, karena kehilangan salah satu file atau kehilangan salah satu isi dalam file yang bersangkutan akan memengaruhi proses desain sistem secara keseluruhan, sehingga pengecekan data sebelum mendesain sistem agar sesuai dengan kebutuhan penelitian bersifat penting dan mandatoris. Hal yang sama juga berlaku dalam instrumen yang akan digunakan. Semua framework yang dibutuhkan harus sudah terinstal terlebih dahulu sebelum prosedur pemodelan akustik.

3.2.3. Desain Sistem

Dalam subbab ini, penulis akan menguraikan gambaran desain sistem secara umum (L. Rabiner, 1989) yang akan dirancang dalam sistem pengenalan ucapan yang menjadi subjek penelitian penulis, sebagai berikut.

1. Input Suara. Suara yang dijadikan masukan direkam secara langsung, kemudian diputar untuk memastikan apakah suara tersebut dapat dibaca di dalam sistem.
2. Ekstraksi fitur. Analisis spektrum dan / atau sinyal suara sementara dilakukan dalam rangka mencari nilai vektor-vektor observasi yang dapat digunakan guna melatih HMMs dengan ciri khas berbagai suara bahasa.
3. Sistem Satuan Pencocokan. Pertama, sebuah pilihan satuan pengenalan ucapan harus dilakukan. Kemungkinan termasuk berdasarkan bahasa satuan sub-kata seperti telepon (atau seperti bagian-bagian telepon), diphones, suku kata demi, suku kata, serta satuan turunan seperti fenemes, fenones, dan satuan akustik. Kemungkinan lain termasuk satuan seluruh kata, dan bahkan satuan yang sesuai dengan kelompok 2 atau kata-kata yang lebih banyak (misalnya, dan, di, dari, dll). Umumnya, semakin sedikit kompleks satuan (misalnya, ponsel), semakin sedikit keberadaan mereka dalam bahasa, dan lebih rumit

(variabel) strukturnya dalam menghasilkan suara yang terus menerus. Untuk pengenalan ucapan dengan kosakata yang besar (melibatkan 1.000 kata atau lebih), penggunaan satuan suara sub-kata hampir diwajibkan karena akan sangat sulit merekam kumpulan pelatihan yang memadai untuk merancang HMMs bagi satuan ukuran kata atau yang lebih besar. Namun, untuk aplikasi khusus (misalnya, kosakata yang sedikit, tugas yang terbatas), kedua hal itu layak dan praktis untuk mempertimbangkan kata sebagai dasar dari satuan suara. Kebebasan dari sebuah satuan dipilih untuk pengenalan, inventarisasi satuan tersebut harus diperoleh melalui pelatihan. Biasanya setiap satuan tersebut ditandai dengan beberapa jenis HMM yang parameternya diperkirakan dari kumpulan pelatihan data suara. Sistem pencocokan satuan menyediakan kemungkinan pencocokan semua urutan satuan pengenalan ucapan menuju ke masukan suara yang tidak diketahui. Beberapa teknik penyediaan skor pencocokan tersebut, dan khususnya untuk menentukan skor pencocokan yang terbaik (terkait batasan leksikal dan sintaksis sistem) meliputi prosedur tumpukan decoding, berbagai bentuk

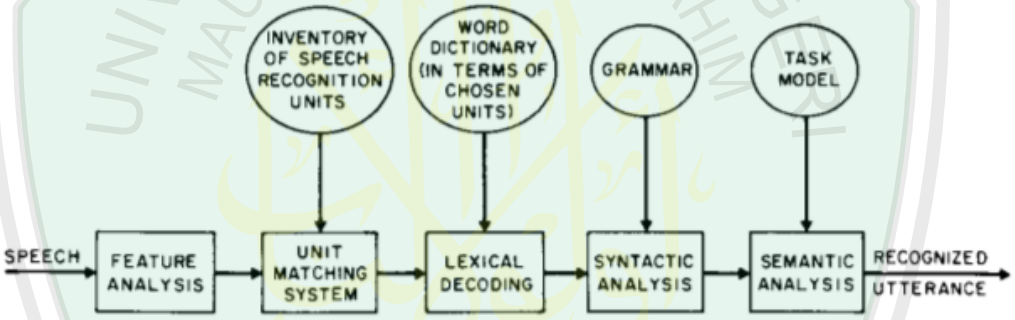
bingkai jalan sinkron decoding, dan prosedur akses penilaian leksikal.

4. Lexical Decoding. Proses ini menempatkan kendala pada sistem satuan pencocokan sehingga jalur yang diselidiki sesuai dengan urutan satuan suara dalam kamus kata (leksikon a). Prosedur ini menyiratkan bahwa kosakata pengenalan ucapan harus ditentukan dalam batas satuan dasar yang dipilih untuk pengenalan tersebut. Spesifikasi tersebut dapat menjadi deterministik (misalnya, satu atau lebih jaringan keadaan (state) yang terbatas untuk setiap kata dalam kosa kata) atau secara statistik (misalnya, probabilitas melekat pada lengkungan dalam representasi keadaan yang terbatas dara beberapa kata). Dalam kasus di mana satuan yang dipilih adalah kata-kata (atau kombinasi kata), langkah decoding leksikal pada dasarnya dihilangkan dan struktur pengenal yang disederhanakan begitu banyak.
5. Analisis Sintak. Proses ini, seperti decoding leksikal, menempatkan kendala lebih lanjut pada sistem satuan pencocokan sehingga jalur diselidiki di mana proses tsb. sesuai dengan satuan suara yang terdiri dari kata (decoding leksikal) dan kata-kata dalam urutan yang tepat seperti yang ditentukan oleh tata bahasa kata (grammar). Seperti

yang demikian, tata bahasa kata dapat diwakili lagi oleh jaringan keadaan deterministik yang terbatas (di mana semua kombinasi kata yang diterima oleh tata bahasa yang disebutkan), atau dengan tata bahasa statistik (misalnya, model kata trigram di mana probabilitas dari rentetan 3 kata dalam urutan tertentu yang diberikan). Untuk beberapa perintah dan kontrol tugas, hanya satu kata yang diperlukan untuk dikenali, oleh karena itu tata bahasa bisa dikatakan sepele atau tidak perlu. Tugas-tugas seperti ini sering disebut tugas pengenalan ucapan dari kata yang terisolasi. Untuk aplikasi lain (misalnya, rentetan digit) tata bahasa yang sangat sederhana sering memadai (misalnya, angka apapun dapat diucapkan dan diikuti oleh digit lainnya). Akhirnya ada tugas di mana tata bahasa merupakan faktor dominan dan, meskipun menambah banyak kendala dalam proses pengenalan, hal itu sangat meningkatkan kinerja pengenalan melalui pembatasan yang dihasilkan pada urutan satuan suara dengan kandidat pengenalan yang benar.

6. Analisa Semantik. Proses ini, sekali lagi seperti langkah-langkah analisis sintaksis dan leksikal decoding, menambahkan kendala lebih lanjut untuk kumpulan jalur-

jalur pencarian pengenalan. Salah satu cara di mana kendala semantik yang digunakan adalah melalui model dinamis dari keadaan pengenalnya (recognizer). Tergantung pada keadaan pengenal, beberapa metode secara sintak membenarkan masukan variabel string dieliminasi dari pertimbangan. Hal ini sekali lagi berfungsi untuk membuat tugas pengenalan menjadi lebih mudah dan mengarah kepada sebuah kinerja sistem yang lebih tinggi.



Gambar 3.1: Alur proses pengenalan ucapan secara umum (L. Rabiner, 1989)

Teknik yang dipakai untuk mengolah data akan dijelaskan berdasarkan komponen yang dibutuhkan, yakni language model, dictionary dan acoustic model.

1. Language Model
    1. Siapkan file txt dengan format yang telah disebutkan pada subbab 3.2.2 bagian Language Model. Pastikan pula bahwa framework CMUCLMTK sebagai instrumen penelitian ini telah diinstal sebelumnya.

Dalam penelitian ini, penulis menggunakan file Hadits_sample.txt dengan isi sebagai berikut

```
<s> اَلْحَيَاءُ لاَ يَأْتِيْ إِلاَّ بِخَيْرٍ </s>
<s> مَنْ سَلِمَ الْمُسْلِمُوْنَ مِنْ لِسَانِهِ وَ يَدِهِ </s>
<s> أَكْمَلُ الْمُؤْمِنِيْنَ إِيْمَانًا أَحْسَنُهُمْ خُلُقًا </s>
<s> إِنَّ اللّهَ جَمِيْلٌ يُحِبُّ الْجَمَالَ </s>
<s> اَلْيَدُ الْعُلْيَا خَيْرٌ مِنْ يَدِ السُّفْلَى </s>
```

2. Buka terminal dan ganti direktori menuju lokasi di mana teks tersebut berada. Lalu buat daftar vocabulary dengan perintah berikut

```
text2wfreq < Hadits_sample.txt | wfreq2vocab > Hadits_sample.vocab
```

Input yang dimaksud dalam perintah di atas adalah Hadits_sample.txt, lalu diubah dalam bentuk file berekstensi .wfreq dengan nama yang sama, kemudian diubah lagi dengan perintah wfreq2vocab dengan struktrur yang tertera dalam perintah tersebut. Hasil eksekusi perintah di atas adalah output berupa Hadits_sample.vocab dengan isi sebagai berikut.

```
## Vocab generated by v2 of the CMU-Cambridge Statistcal
## Language Modeling toolkit.
##
## Includes 29 words ##
</s>
<s>
أَحْسَنُهُمْ
```

3. Buat language model berformat ARPA (.lm) dengan mengetikkan perintah berikut

```
1. text2idngram -vocab Hadits_sample.vocab -idngram
   Hadits_sample.idngram < Hadits_sample.txt
2. idngram2lm -vocab_type 0 -idngram Hadits_sample.idngram
   -vocab  Hadits_sample.vocab -arpa Hadits_sample.lm
```

Input pada perintah pertama adalah vocabulary yang diambil dari berkas Hadits_sample.txt, sebagaimana dieksekusi melalui perintah sebelumnya. Output perintah pertama adalah Hadits_sample.idngram yang berbentuk binary.

Input pada perintah kedua berupa idngram dan vocabulary, sedangkan output perintah di atas berbentuk file berekstensi lm yang berbentuk teks dengan isi sebagai berikut.

```
################################################################
################### Copyright (c) 1996, Carnegie Mellon University, Cambridge University,
## Ronald Rosenfeld and Philip Clarkson
## Version 3, Copyright (c) 2006, Carnegie Mellon University
## Contributors includes Wen Xu, Ananlada Chotimongkol,
## David Huggins-Daines, Arthur Chan and Alan Black
################################################################
#################
================================================================
============== This file was produced by the CMU-Cambridge ==============
============== Statistical Language Modeling Toolkit ==============
================================================================
============================
This is a 3-gram language model, based on a vocabulary of 29 words,
  which begins "</s>", "<s>", "أَحْسَنُهُمْ"...
This is a CLOSED-vocabulary model
  (OOVs eliminated from training data and are forbidden in test data)
Good-Turing discounting was applied.
1-gram frequency of frequency : 25
2-gram frequency of frequency : 32 0 0 1 0 0 0
3-gram frequency of frequency : 36 0 0 0 0 0 0
1-gram discounting ratios : 0.96
```

2-gram discounting ratios :
3-gram discounting ratios :
This file is in the ARPA-standard format introduced by Doug Paul.

p(wd3|wd1,wd2) = if (trigram exists) p_3(wd1,wd2,wd3)
else if (bigram w1,w2 exists) bo_wt_2(w1,w2)*p(wd3|wd2)
else p(wd3|w2)

p(wd2|wd1)= if (bigram exists) p_2(wd1,wd2)
else bo_wt_1(wd1)*p_1(wd2)

All probs and back-off weights (bo_wt) are given in log10 form.

Data formats:

Beginning of data mark: \data\
ngram 1=nr # number of 1-grams
ngram 2=nr # number of 2-grams
ngram 3=nr # number of 3-grams

\1-grams:
p_1 wd_1 bo_wt_1
\2-grams:
p_2 wd_1 wd_2 bo_wt_2
\3-grams:
p_3 wd_1 wd_2 wd_3

end of data mark: \end\

\data\
ngram 1=29
ngram 2=33
ngram 3=36

\1-grams:
-0.9542 </s> -0.7132
-0.8573 <s> -0.9791
-1.5733 أَحْسَنُهُمْ 0.4654-
-1.5733 أَكْمَلُ 0.4654-
-1.5733 إِنَّ 0.4654-
-1.5733 إِيْمَانًا 0.4654-
-1.5733 إِلَّا 0.4654-
-1.5733 السُّفْلَى 0.0000
-1.5733 اللَّهَ 0.4654-

-1.5733 الْجَمَالَ 0.4260-
-1.5733 الْعُلْيَا 0.4654-
-1.5733 الْمُؤْمِنِيْنَ 0.4654-
-1.5733 الْمُسْلِمُوْنَ 0.4523-
-1.5733 اَلْحَيَاءُ 0.4654-
-1.5733 اَلْيَدُ 0.4654-
-1.5733 بِخَيْرٍ 0.4260-
-1.5733 جَمِيْلٌ 0.4654-
-1.5733 خَيْرٌ 0.4523-
-1.5733 خُلُقًا 0.4260-
-1.5733 سَلِمَ 0.4654-
-1.5733 لِسَانِهِ 0.4654-
-1.5733 مَنْ 0.4654-
-1.2553 مِنْ 0.6751-
-1.5733 وَ 0.4654-
-1.5733 يَأْتِيْ 0.4654-
-1.5733 يَدِ 0.4654-
-1.5733 يَدِهِ 0.4260-
-1.5733 يُحِبُّ 0.4654-
-1.5733 لاَ 0.4654-

\2-grams:
-0.0792 </s> <s> -0.1347
-0.7404 <s> 0.1761 أَكْمَلُ
-0.7404 <s> 0.1761 إِنَّ
-0.7404 <s> 0.1761 اَلْحَيَاءُ
-0.7404 <s> 0.1761 اَلْيَدُ
-0.7404 <s> 0.1761 مَنْ
-0.1761 0.1761 أَحْسَنُهُمْ خُلُقًا
-0.1761 0.1761 أَكْمَلُ الْمُؤْمِنِيْنَ
-0.1761 0.1761 إِنَّ اللهَ
-0.1761 0.1761 إِيْمَانًا أَحْسَنُهُمْ
-0.1761 0.1761 إِلاَّ بِخَيْرٍ
-0.1761 0.1761 اللهَ جَمِيْلٌ
-0.1761 الْجَمَالَ </s> 0.4771
-0.1761 0.1761 الْعُلْيَا خَيْرٌ
-0.1761 0.1761 الْمُؤْمِنِيْنَ إِيْمَانًا
-0.1761 0.0792- الْمُسْلِمُوْنَ مِنْ
-0.1761 0.1761 اَلْحَيَاءُ لاَ
-0.1761 0.1761 اَلْيَدُ الْعُلْيَا
-0.1761 بِخَيْرٍ </s> 0.4771
-0.1761 0.1761 جَمِيْلٌ يُحِبُّ
-0.1761 0.0792- خَيْرٌ مِنْ
-0.1761 خُلُقًا </s> 0.4771

-0.1761 0.1761 سَلِمَ الْمُسْلِمُوْنَ
-0.1761 0.1761 لِسَانِهِ وَ
-0.1761 0.1761 مَنْ سَلِمَ
-0.3979 0.1761 مِنْ لِسَانِهِ
-0.3979 0.1761 مِنْ يَدِ
-0.1761 0.1761 وَ يَدِهِ
-0.1761 0.1761 يَأْتِيْ إِلَّا
-0.1761 0.2499- يَدِ السُّفْلَى
-0.1761 0.4771 يَدِهِ </s>
-0.1761 0.1761 يُحِبُّ الْجَمَالَ
-0.1761 0.1761 لَا يَأْتِيْ

\3-grams:
-0.6990 أَكْمَلُ </s> <s>
-0.6990 إِنَّ </s> <s>
-0.6990 اَلْيَدُ </s> <s>
-0.6990 مَنْ </s> <s>
-0.3010 أَكْمَلُ الْمُؤْمِنِيْنَ <s>
-0.3010 إِنَّ اللّٰهَ <s>
-0.3010 اَلْحَيَاءُ لَا <s>
-0.3010 اَلْيَدُ الْعُلْيَا <s>
-0.3010 مَنْ سَلِمَ <s>
-0.3010 أَحْسَنُهُمْ خُلُقًا </s>
-0.3010 أَكْمَلُ الْمُؤْمِنِيْنَ إِيْمَانًا
-0.3010 إِنَّ اللّٰهَ جَمِيْلٌ
-0.3010 إِيْمَانًا أَحْسَنُهُمْ خُلُقًا
-0.3010 إِلَّا بِخَيْرٍ </s>
-0.3010 اللّٰهَ جَمِيْلٌ يُحِبُّ
-0.3010 الْجَمَالَ </s> <s>
-0.3010 الْعُلْيَا خَيْرٌ مِنْ
-0.3010 الْمُؤْمِنِيْنَ إِيْمَانًا أَحْسَنُهُمْ
-0.3010 الْمُسْلِمُوْنَ مِنْ لِسَانِهِ
-0.3010 اَلْحَيَاءُ لَا يَأْتِيْ
-0.3010 اَلْيَدُ الْعُلْيَا خَيْرٌ
-0.3010 بِخَيْرٍ </s> <s>
-0.3010 جَمِيْلٌ يُحِبُّ الْجَمَالَ
-0.3010 خَيْرٌ مِنْ يَدِ
-0.3010 خُلُقًا </s> <s>
-0.3010 سَلِمَ الْمُسْلِمُوْنَ مِنْ
-0.3010 لِسَانِهِ وَ يَدِهِ
-0.3010 مَنْ سَلِمَ الْمُسْلِمُوْنَ
-0.3010 مِنْ لِسَانِهِ وَ
-0.3010 مِنْ يَدِ السُّفْلَى
-0.3010 وَ يَدِهِ </s>

```
-0.3010 يَأْتِيْ إِلاَّ بِخَيْرٍ
-0.3010 يَدِ السُّفْلَى </s>
-0.3010 يَدِهِ </s> <s>
-0.3010 يُحِبُّ الْجَمَالَ </s>
-0.3010 لاَ يَأْتِيْ إِلاَّ

\end\
```

Parameter -vocab_type 0 menunjukkan bahwa kosa kata yang diberikan bersifat tertutup. Hal tersebut dapat ditemukan pada isi format language model yang telah dihasilkan. Artinya tipe kosa kata tersebut tidak mengizinkan adanya kata di luar daftar yang sudah ada. Dengan demikian, kata apa pun di luar daftar kata yang diberikan akan menyebabkan error dalam proses data training dan testing.

4. Buat format binary (DMP) dari language model yang dihasilkan dengan perintah berikut

```
sphinx_lm_convert -i Hadits_sample.lm -o Hadits_sample.DMP
```

Perintah di atas merupakan perintah konversi dengan input dari Hadits_sample.lm dan dubah sebagai output menjadi Hadits_sample.lm.DMP. Selain bentuk DMP, format bin didukung oleh perintah ini. Perintah tersebut dapat berlaku sebaliknya, yakni mengubah file dengan

tipe binary ke dalam bentuk teks. Namun, pastikan bahwa framework sphinxbase telah diinstal sebelumnya agar dapat mengeksekusi perintah di atas.

Bila ingin mengetahui lebih lanjut semua perintah di atas beserta kegunaannya, maka pengguna dapat merujuk pada file dokumentasi yang tertera di dalam folder framework.

Sedangkan file berekstensi jsgf telah disiapkan penulis secara manual sesuai isi file vocabulary yang telah dihasilkan dari teks sebelumnya. Berikut ini merupakan isi file tersebut

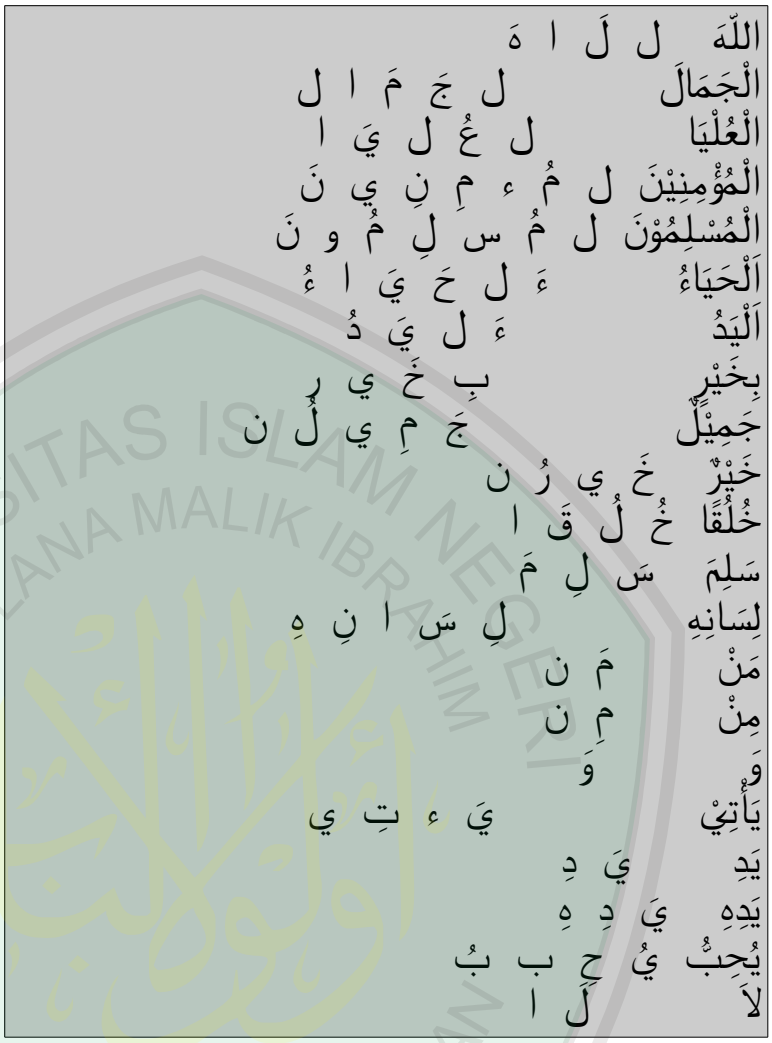
```
#JSGF V1.0 UTF-8;

grammar hadits_1;

public <word_01> = أَحْسَنُهُمْ;
public <word_02> = أَكْمَلُ;
public <word_03> = إِنَّ;
public <word_04> = إِيْمَانًا;
public <word_05> = إِلَّا;
public <word_06> = السُّفْلَى;
public <word_07> = اللهِ;
public <word_08> = الْجَمَالَ;
public <word_09> = الْعُلْيَا;
public <word_10> = الْمُؤْمِنِيْنَ;
public <word_11> = الْمُسْلِمُوْنَ;
public <word_12> = اَلْحَيَاءُ;
public <word_13> = اَلْيَدُ;
public <word_14> = بِخَيْرٍ;
public <word_15> = جَمِيْلٌ;
public <word_16> = خَيْرٌ;
public <word_17> = خُلُقًا;
public <word_18> = سَلِمَ;
public <word_19> = لِسَانِهِ;
public <word_20> = مَنْ;
```

```
public <word_21> = مِنْ;
public <word_22> = وَ;
public <word_23> = يَأْتِيْ;
public <word_24> = يَدِ;
public <word_25> = يَدِهِ;
public <word_26> = يُحِبُّ;
public <word_27> = لَا;

public <sentence_01> = <word_12> <word_27> <word_23> <word_05>
<word_14>;
public <sentence_02> = <word_20> <word_18> <word_11> <word_21>
<word_19> <word_22> <word_25>;
public <sentence_03> = <word_02> <word_10> <word_04> <word_01>
<word_17>;
public <sentence_04> = <word_03> <word_07> <word_15> <word_26>
<word_08>;
public <sentence_05> = <word_13> <word_09> <word_16> <word_21>
<word_24> <word_06>;

<answer> = (<sentence_01> | <sentence_02> | <sentence_03> |
<sentence_04> | <sentence_05>);
```

2. Dictionary

   Seperti yang dijelaskan di muka, penulis membuat file berekstensi .dic yang memuat daftar kata beserta pengucapannya secara manual. Tata cara penulisan mengikuti format yang telah dipaparkan pada subbab sebelumnya. Hasilnya dapat dilihat berikut ini.

```
أَحْسَنُهُمْ    ءَ ح سَ نُ هُ م
أَكْمَلُ    ءَ كْ مَ لُ
إِنَّ    ءِ ن نَ
إِيْمَانًا    ءِ ي مَ ا نَ ن
إِلَّا    ءِ ل لَ ا
أَلسُّفْلَى    ا س سُ فـ لَ ا
```



3. Acoustic Model

Selain Language Model dan Dictionary, berikut ini merupakan file yang harus dipersiapkan sebelum membuat model akustik

1. Filler Dictionary. File ini memuat suara yang dibuat dalam bentuk teks, namun tidak terdapat dalam kamus

pengucapan (Pronunciation Dictionary). Dalam penelitian ini, penulis hanya mencakup tiga daftar suara

```
<s>    SIL
</s>   SIL
<sil>  SIL
```

Tag di atas merupakan representasi dari suara tanpa kata (silence). Sebenarnya masih ada berbagai suara yang dapat dimasukkan ke dalam file ini (seperti suara bersin, melamun, batuk, dan berbagai suara bising lainnya), namun penulis cukupkan di tiga tanda ini.

2. Phonetic Dictionary. File ini berisi fonem yang akan dipakai dalam dictionary dan berkas transcription, baik untuk keperluan training maupun testing. Penulis membuat daftar fonem yang akan dipakai dalam pembuatan model akustik dengan bentuk berikut

ا ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي

```
يُ
ا
```

Penulisan fonem di atas harus ada di dalam file yang memanfaatkan daftar fonem ini, baik dictionary maupun transcription. Karena itu, framework Sphinx tidak mengizinkan ada fonem yang tidak terdapat dalam berkas-berkas yang membutuhkannya, yaitu dictionary dan transcription. Sebaliknya, sphinx juga tidak mengizinkan penggunaan fonem, baik dalam dictionary maupun transcription, yang berada di luar daftar fonem yang dipakai.

3. Berkas fileids. Berkas ini berbentuk teks yang dibuat secara manual dengan format yang telah dijelaskan di muka. Berikut ini merupakan isi berkas fileids yang dibuat penulis.

```
speaker_001/Hadits_sample_0001
speaker_001/Hadits_sample_0002
speaker_001/Hadits_sample_0003
speaker_001/Hadits_sample_0004
speaker_001/Hadits_sample_0005
speaker_002/Hadits_sample_0001
speaker_002/Hadits_sample_0002
speaker_002/Hadits_sample_0003
speaker_002/Hadits_sample_0004
speaker_002/Hadits_sample_0005
…
speaker_160/Hadits_sample_0001
speaker_160/Hadits_sample_0002
```

```
speaker_160/Hadits_sample_0003
speaker_160/Hadits_sample_0004
speaker_160/Hadits_sample_0005
```

Berkas suara berekstensi wav harus tersedia dalam folder terpisah dengan folder yang mengandung teks sebagai pra-pemrosesan model akustik (dictionary, fonem, filler, transcription, fileids, jsgf, lm.DMP ada di dalam folder etc). Dalam penelitian ini, penulis menempatkan berkas suara Hadits_sample_0001.wav hingga Hadits_sample_0005.wav dalam sub-folder speaker_001 hingga speaker_160, yang terdapat dalam folder wav. Isi tersebut berlaku hanya untuk kepentingan training, sehingga jumlah baris dalam fileids untuk training berjumlah 800. Khusus untuk testing, cukup lima baris pertama yang terkandung di dalamnya.

4. Berkas transcription. Berkas ini memuat kalimat yang diawali dengan tanda <s>, diakhiri dengan tanda </s>, dan ditambahkan dengan nama berkas suara dalam kurung setelah tanda </s>. Dalam penelitian ini, penulis memaparkan isi data tersebut dalam bentuk teks berikut

```
<s> اَلْحَيَاءُ لَا يَأْتِيْ إِلَّا بِخَيْرٍ </s> (Hadits_sample_0001)
<s> مَنْ سَلِمَ الْمُسْلِمُوْنَ مِنْ لِسَانِهِ وَ يَدِهِ </s> (Hadits_sample_0002)
```

<s> أَكْمَلُ الْمُؤْمِنِيْنَ إِيْمَانًا أَحْسَنُهُمْ خُلُقًا </s> (Hadits_sample_0003)
<s> إِنَّ اللّٰهَ جَمِيْلٌ يُحِبُّ الْجَمَالَ </s> (Hadits_sample_0004)
<s> اَلْيَدُ الْعُلْيَا خَيْرٌ مِنْ يَدِ السُّفْلَى </s> (Hadits_sample_0005)

seperti format yang dijelaskan dalam berkas fileids, semua berkas suara yang terkandung dalam subfolder, (Hadits_sample_0001.wav, Hadits_sample_0002.wav, Hadits_sample_0003.wav., Hadits_sample_0004.wav, dan Hadits_sample_0005.wav) harus terdapat dalam satu subfolder yang termasuk dalam folder wav. Penulis menempatkan kelimanya dalam subfolder speaker_001 hingga subfolder speaker_160, seperti yang telah dijelaskan dalam poin sebelumnya. Khusus untuk training, kelima baris di atas diulang sebanyak 160 kali sehingga total baris dalam file transcription sebanyak 800 baris, setara dengan jumlah baris dalam fileids. Sedangkan untuk testing, kelima baris di atas tidak perlu diulang.

5. Berkas suara dengan format wav. Berkas ini memiliki sample rate 16000 Hz, Bit Rate 16 Bit, dengan tipe mono. Dalam penelitian ini, semuanya berada dalam subfolder speaker_001 hingga subfolder speaker_160

Dengan demikian, struktur folder beserta file yang telah disediakan akan berbentuk seperti ini

- etc
  - *Hadits_sample*.dic
  - *Hadits_sample*.filler
  - *Hadits_sample*.jsgf
  - *Hadits_sample*.lm.DMP
  - *Hadits_sample*.phone
  - *Hadits_sample_test*.fileids
  - *Hadits_sample_test*.transcription
  - *Hadits_sample_train*.fileids
  - *Hadits_sample_train*.transcription
- wav
  - speaker_001
    - *Hadits_sample*_0001.wav
    - *Hadits_sample*_0002.wav
    - *Hadits_sample*_0003.wav
    - *Hadits_sample*_0004.wav
    - *Hadits_sample*_0005.wav
  - speaker_002
    - *Hadits_sample*_0001.wav
    - *Hadits_sample*_0002.wav
    - *Hadits_sample*_0003.wav
    - *Hadits_sample*_0004.wav
    - *Hadits_sample*_0005.wav

    ...
  - speaker_160
    - *Hadits_sample*_0001.wav
    - *Hadits_sample*_0002.wav
    - *Hadits_sample*_0003.wav
    - *Hadits_sample*_0004.wav
    - *Hadits_sample*_0005.wav

Setelah semua file pendukung sudah disiapkan untuk pelatihan model akustik dan framework bantu sudah terinstal, penulis mulai memasuki proses pembuatan model akustik dengan langkah-langkah sebagai berikut.

1. Buka terminal, lalu arahkan direktori menuju folder database tempat semua berkas disiapkan. Ketikkan perintah berikut

```
sphinxtrain -t Hadits_sample setup
```

Lalu tekan enter untuk membuat database. Perintah di atas menggunakan database bernama Hadits_sample, sesuai dengan nama file yang telah disiapkan, untuk kemudahan. Setelah perintah tersebut dieksekusi, dua folder akan terbentuk di dalam folder Hadits_sample, yaitu etc dan wav. Di samping itu, berkas sphinx_train.cfg akan terbentuk di dalam folder etc.

2. Buka file sphinx_train.cfg untuk memulai proses konfigurasi sebelum training.

   a) Konfigurasi berkas audio. Karena penulis menggunakan berkas audio dengan format wav, maka konfigurasi yang sesuai adalah sebagai berikut

```
$CFG_WAVFILES_DIR = "$CFG_BASE_DIR/wav";
$CFG_WAVFILE_EXTENSION = 'wav';
$CFG_WAVFILE_TYPE = 'mswav'; # one of nist, mswav, raw
```

   b) Konfigurasi parameter path menuju berkas yang dibutuhkan. Variabel yang digunakan adalah lokasi

dictionary, phoneme, filler, transcription, dan fileids. Parameter ini berisi informasi tentang berkas yang dipakai untuk proses training.

```
# Variables used in main training of models
$CFG_DICTIONARY = "$CFG_LIST_DIR/$CFG_DB_NAME.dic";
$CFG_RAWPHONEFILE = "$CFG_LIST_DIR/$CFG_DB_NAME.phone";
$CFG_FILLERDICT = "$CFG_LIST_DIR/$CFG_DB_NAME.filler";
$CFG_LISTOFFILES = "$CFG_LIST_DIR/${CFG_DB_NAME}_train.fileids";
$CFG_TRANSCRIPTFILE = "$CFG_LIST_DIR/${CFG_DB_NAME}_train.transcription"
```

Karena nama file yang dibutuhkan beserta lokasinya telah penulis tetapkan sejak awal, variabel di atas tidak perlu diubah. Variabel $CFG_LIST_DIR secara bawaan adalah folder etc dalam database yang telah dibuat, sedangkan variabel $CFG_DB_NAME adalah nama project yang diwakili oleh nama file.

c) Konfigurasi tipe model akustik dan parameter model. Sphinx memiliki tiga tipe model akustik yang akan dibuat: continous, semi-continous, and Phonetically Tied Method (PTM).

```
$CFG_HMM_TYPE = '.cont.'; # Sphinx4, Pocketsphinx
#$CFG_HMM_TYPE  = '.semi.'; # PocketSphinx only
#$CFG_HMM_TYPE  = '.ptm.'; # Sphinx4, Pocketsphinx, faster model
```

Cukup hilangkan tanda pagar untuk menghapus komentar yang mendahului variabel yang akan digunakan. Penulis menggunakan tipe continous dengan pertimbangan akurasi pengenalan dan data yang akan ditampung dalam proses komputasi.

```
$ $CFG_FINAL_NUM_DENSITIES = 8;
# Number of tied states (senones) to create in decision-tree clustering
$CFG_N_TIED_STATES = 200;
$CFG_CD_TRAIN = 'yes;
```

Baris pada perintah pertama menunjukkan banyaknya densitas akhir dalam memproses parameter yang dipakai dalam membentuk model. Penulis menetapkan nilai 8 sesuai nilai bawaan file. Baris kedua adalah parameter untuk menentukan banyaknya senone yang dipakai untuk clustering. Penulis membatasinya pada nilai 200, menyesuaikan banyaknya data suara yang dipakai sebagai acuan. Intinya, semakin tinggi nilai senone, framework akan semakin jeli dalam membedakan berbagai suara yang dijadikan acuan dalam training. Baris ketiga adalah variabel yang meminta konfirmasi apakah menggunakan model yang

bergantung konteks. Penulis mengisinya dengan nilai 'yes', yang berarti model yang dihasilkan dapat terikat konteks dan bebas konteks.

d) Konfigurasi parameter fitur suara. Seperti yang dijelaskan di muka, penulis menggunakan data suara dengan sample rate 16000 Hz, Bit Rate 16 Bit, dan tipe channel mono. Secara bawaan, sphinx menggunakan konfigurasi standar dengan data suara yang sama dengan data yang disiapkan penulis, sehingga penulis tidak melakukan perubahan apa pun terhadap konfigurasi yang ada.

```
# Feature extraction parameters
$CFG_WAVFILE_SRATE = 16000.0;
$CFG_NUM_FILT = 25; # For wideband speech it's 25, for telephone 8khz
reasonable value is 15
$CFG_LO_FILT = 130; # For telephone 8kHz speech value is 200
$CFG_HI_FILT = 6800; # For telephone 8kHz speech value is 3500
```

e) Konfigurasi pemrosesan paralel. Hal ini bertujuan meningkatkan performa pada saat training data. Dalam kasus ini, penulis mengubah nilai parameter $CFG_QUEUE_TYPE menjadi ”QUEUE::POSIX” karena komputer yang dipakai menggunakan lebih dari 1 core CPU.

```
# Queue::POSIX for multiple CPUs on a local machine
 # Queue::PBS to use a PBS/TORQUE queue
$CFG_QUEUE_TYPE = "Queue::POSIX";

# How many parts to run Forward-Backward estimatinon in
$CFG_NPART = 1;
$DEC_CFG_NPART = 1; #  Define how many pieces to split decode in
```

Sedangkan dua variabel terakhir tetap bernilai 1, sesuai bawaan dari berkas konfigurasi yang dipakai.

f) Konfigurasi parameter decoding. Konfigurasi ini kurang lebih sama dengan konfigurasi parameter path dengan berkas yang dibutuhkan. Perbedaan keduanya terletak pada keperluannya. Parameter decoding menggunakan path untuk keperluan testing.

```
$DEC_CFG_DICTIONARY = "$DEC_CFG_BASE_DIR/etc/$DEC_CFG_DB_NAME.dic";
$DEC_CFG_FILLERDICT = "$DEC_CFG_BASE_DIR/etc/$DEC_CFG_DB_NAME.filler";
$DEC_CFG_LISTOFFILES = "$DEC_CFG_BASE_DIR/etc/$
{DEC_CFG_DB_NAME}_test.fileids";
$DEC_CFG_TRANSCRIPTFILE = "$DEC_CFG_BASE_DIR/etc/$
{DEC_CFG_DB_NAME}_test.transcription";
$DEC_CFG_RESULT_DIR = "$DEC_CFG_BASE_DIR/result";

# These variables, used by the decoder, have to be user defined, and
# may affect the decoder output
```

```
$DEC_CFG_LANGUAGEMODEL_DIR = "$DEC_CFG_BASE_DIR/etc";
$DEC_CFG_LANGUAGEMODEL =
"$DEC_CFG_LANGUAGEMODEL_DIR/an4.lm.DMP";
$DEC_CFG_GRAMMAR  = "$CFG_BASE_DIR/etc/${CFG_DB_NAME}.jsgf";
```

Namun karena testing tidak dilakukan dalam proses ini, konfigurasi di atas tidak diubah sama sekali, kecuali parameter terakhir. Penulis menambahkan variabel $DEC_CFG_GRAMMAR secara opsional, tergantung tingkat kesalahan dalam training.

Setelah konfigurasi selesai dibuat, proses selanjutnya adalah proses training

3. Kembali ke teminal dengan direktori yang sama dengan langkah pertama. Ketikkan perintah berikut

```
sphinxbase run
```

Lalu tekan enter untuk melakukan proses training

Selesai menjalani proses training, isi database yang telah dihasilkan akan berbentuk susunan folder seperti ini

```
etc
feat
logdir
model_parameters
model_architecture
result
wav
```

Sedangkan hasil pemodelan akustik yang telah dilalui penulis, terletak pada direktori di bawah ini

```
model_parameters/<nama_db>.cd_cont_<number_of senones>
```

Nama_db yang ada dalam folder model_parameter bernama Hadits_sample, sesuai dengan nama database yang telah ditetapkan penulis sebelumnya. Dengan demikian, nama subfolder yang telah penulis hasilkan dalam folder induk model_parameters adalah Hadits_sample.cd_cont_200, di samping model bebas konteks bernama Hadits_sample.ci_cont.

Selanjutnya, isi folder yang telah dibuat untuk model akustik adalah berikut ini

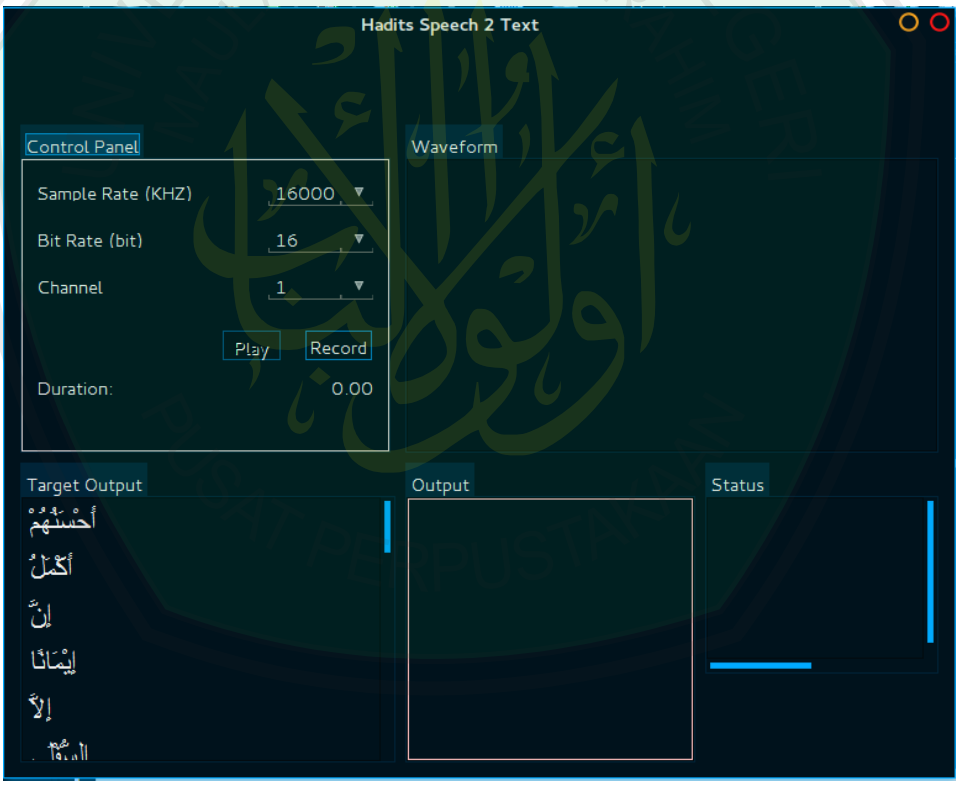
```
mdef
feat.params
mixture_weights
means
noisedict
transition_matrices
variances
```

Dengan demikian, proses pembuatan model akustik telah selesai, dan folder berisi komponen model akustik telah siap digunakan.

Perintah di atas juga berfungsi untuk menjalankan perintah testing, asalkan jumlah durasi total data suara yang dibutuhkan telah mencukupi, minimal satu jam.

3.2.4. Perancangan dan Pembuatan Aplikasi

Untuk membuat aplikasi pengenalan ucapan menjadi interaktif, penulis menawarkan antarmuka yang ditunjukkan dalam gambar 3.2.



Gambar 3.2: Contoh antarmuka aplikasi pengenalan ucapan

Berikut ini adalah tabel penjelasan masing-masing komponen dalam Antarmuka aplikasi pengenalan ucapan

No	Komponen	Tipe	Penjelasan

1	cb_sample	Combo Box	Input sample Rate berupa combo box. Penulis membatasi hanya satu pilihan nilai, yakni 16000. Nilai tersebut akan dikonversi ke dalam tipe data float untuk proses lebih lanjut
2	cb_bit	Combo Box	Input bit Rate berupa combo box. Nilai yang dicakup dalam aplikasi tersebut hanya 1, yakni 16. Nilai tersebut akan dikonversi ke dalam tipe data integer untuk proses lebih lanjut
3	cb_channel	Combo Box	Input channel Rate berupa combo box. Nilai yang terdapat dalam komponen ini adalah 1 (mono). Input ini berpengaruh pada jumlah spektrum yang akan ditampilkan setelah data suara berhasil direkam. Nilai tersebut akan dikonversi ke dalam tipe data integer untuk proses lebih lanjut
4	val_length	Label	Penanda durasi data suara.

			Secara bawaan, komponen tersebut memiliki format ' – : – : –'. Label ini memiliki format yang sama ketika berada dalam proses menulis dan membaca data suara. Setelah proses menulis dan membaca data suara (record and play) berhenti, format timer berubah menjadi '– hr – min – sec'
6	btn_play	Tombol (Button)	Memainkan suara yang telah direkam. Secara bawaan, tombol ini bersifat nonaktif sampai ada suara yang berhasil direkam. Setelah direkam dan tombol btn_play ditekan, timer akan berjalan dan label val_length berubah format menjadi ' – : – : –' hingga akhir data suara, di samping proses pembacaan data suara. Proses pembacaan data suara dapat dihentikan di tengah

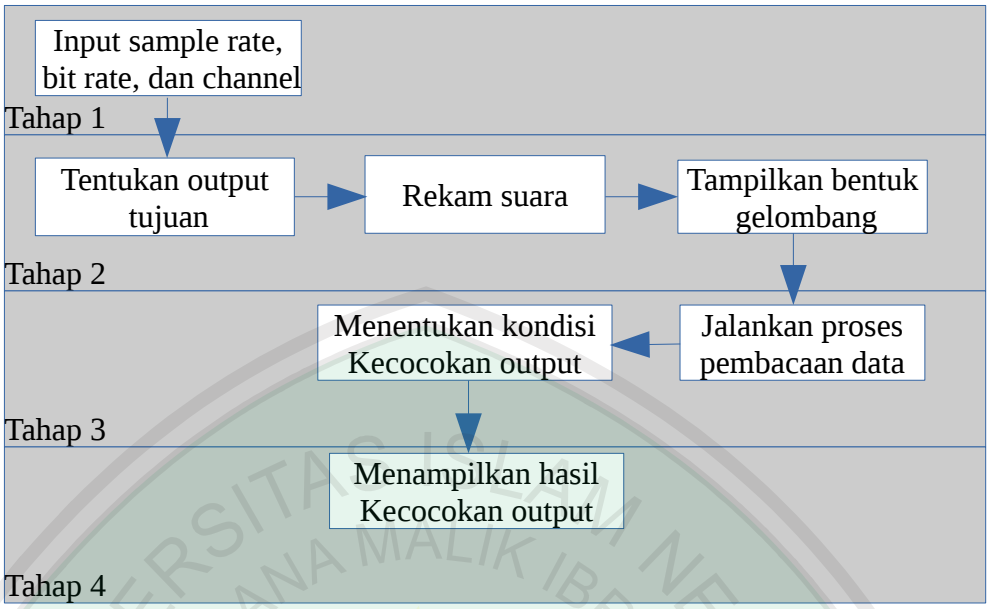
			proses dengan menekan kembali btn_play yang bertuliskan Stop. Setelah proses berhenti, label val_length menampilkan durasi hasil perekaman suara dalam format '– hr – min – sec'
7	btn_rec	Tombol (Button)	Merekam suara dari mikrofon. Setelah ditekan, timer akan berjalan dan label val_length berubah format menjadi ' – : – : –', di samping masuk ke dalam proses perekaman suara, sampai tombol btn_rec yang bertuliskan Stop ditekan. Setelah proses berhenti, label val_length menampilkan durasi hasil perekaman suara dalam format '– hr – min – sec'
8	pnl_ctrl	Panel	Panel yang mencakup input (Sample Rate, Bit Rate, dan Channel) dan aksi (Save, Play, dan Record)
9	pnl_wave	Panel	Menampilkan spektrum sebagai

			data suara secara visual. Spektrum tersebut muncul setelah data suara berhasil diinputkan
10	ta_result	Text Area	Menampilkan hasil pengenalan ucapan berupa teks
11	list_target	List	Memuat daftar output yang akan dijadikan perbandingan terhadap output hasil pengenalan ucapan
12	ta_stat	Text Area	Sebagai penanda kecocokan antara output target dan output hasil pengenalan ucapan.

Tabel 3.1: Komponen antarmuka yang digunakan

3.2.5. Perancangan Alur Program

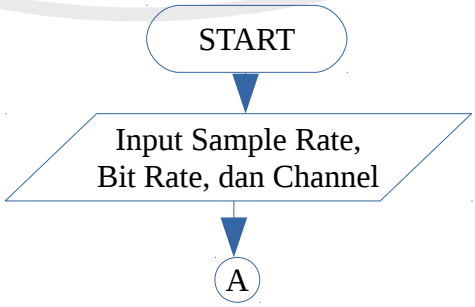
Tahap ini menjelaskan mekanisme berjalannya sistem pengenalan ucapan melalui program yang telah penulis buat. Tahap yang diwakili oleh gambar 3.3 dijalani setelah basis pengetahuan (*linguist*) telah ditentukan melalui proses training dan testing, sebagaimana telah dijelaskan di muka.



Gambar 3.3: Diagram Blok Sistem

- **TAHAP 1**

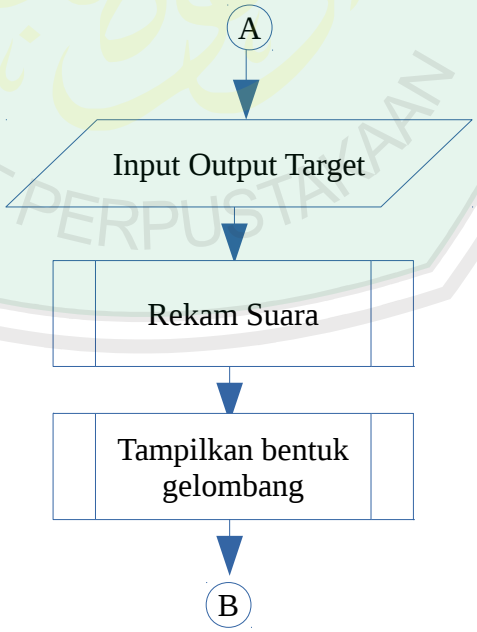
Tahap pertama hanya terdiri dari input sample rate, bit rate, dan channel. Sample rate adalah jumlah gelombang yang dijadikan sample dalam satu satuan waktu. Bit rate adalah jumlah bit rata-rata yang dihitung dalam komputasi gelombang suara. Sedangkan channel adalah tipe saluran yang dipakai untuk mengolah suara. Ketiga variabel tersebut dibuat seragam untuk percobaan yang akan dijelaskan pada bab 4.



Gambar 3.4: Flowchart Input variabel sample rate, bit rate, dan channel

- **TAHAP 2**

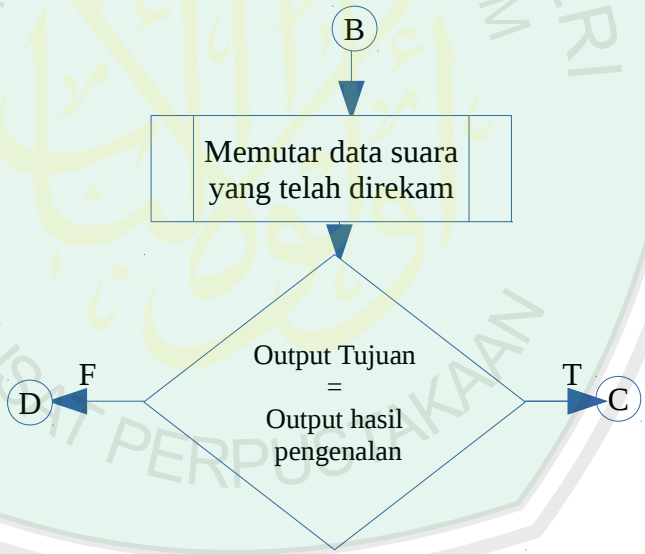
Tahap ini dibagi menjadi tiga langkah. Pertama, tentukan output tujuan yang diharapkan. Output yang dimaksud berupa daftar kata yang telah dibuat pada saat proses pemodelan bahasa (*Language Model*). Kedua, lanjut ke proses perekaman. Begitu, proses merekam dimulai, program memulai proses pengenalan ucapan. Selama proses merekam belum dihentikan, proses perekaman akan terus berjalan. Setelah suara yang diinputkan telah cukup, proses rekaman dapat dihentikan, yang juga mengakhiri proses pengenalan ucapan. Kemudian, bentuk gelombang ditampilkan sebagai bentuk visualisasi data audio yang telah diinputkan.



Gambar 3.5: Flowchart Tahap 2

- **TAHAP 3**

Tahap berikutnya terdiri dari dua langkah, yakni pembacaan data suara dan menentukan kondisi kecocokan. Pembacaan data dalam penelitian ini berupa pemutaran audio yang telah direkam, sambil menjalankan timer. Setelah pembacaan selesai, program menampilkan hasil pengenalan ucapan dari audio tersebut. Kata yang keluar tergantung keberhasilan program dalam mengenali audio yang akan diubah dalam bentuk teks yang diinginkan

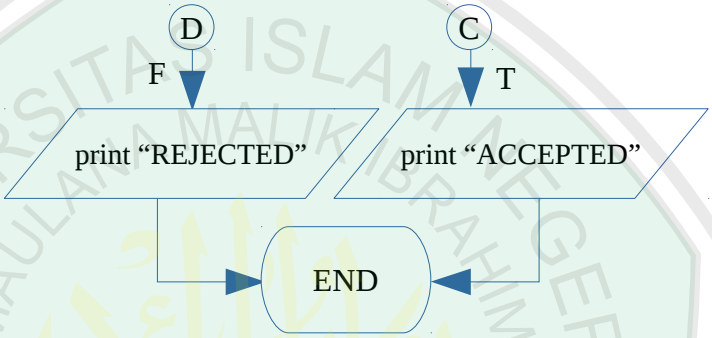


Gambar 3.6: Flowchart Tahap 3

- **TAHAP 4**

Tahap ini merupakan tahap terakhir dalam struktur diagram blok sistem. Jika output tujuan berbeda dengan output hasil

pengenalan, maka program akan menampilkan status "REJECTED". Artinya, hasil pengenalan ditolak. Jika output tujuan sama dengan output hasil pengenalan, maka program akan menampilkan status "ACCEPTED", yang berarti hasil pengenalan diterima.



Gambar 3.7: Flowchart Tahap 4

3.2.6. Uji Coba dan Evaluasi

Prosedur ini dilakukan setelah aplikasi dan sistem telah terintegrasi. Di samping itu, tahap ini mencakup analisa kelebihan dan kekurangan aplikasi yang telah dibangun beserta akurasi metode yang diterapkan dalam sistem, sehingga dapat diketahui apakah hasil proses pengenalan ucapan memunculkan teks yang diinginkan pembicara atau tidak.

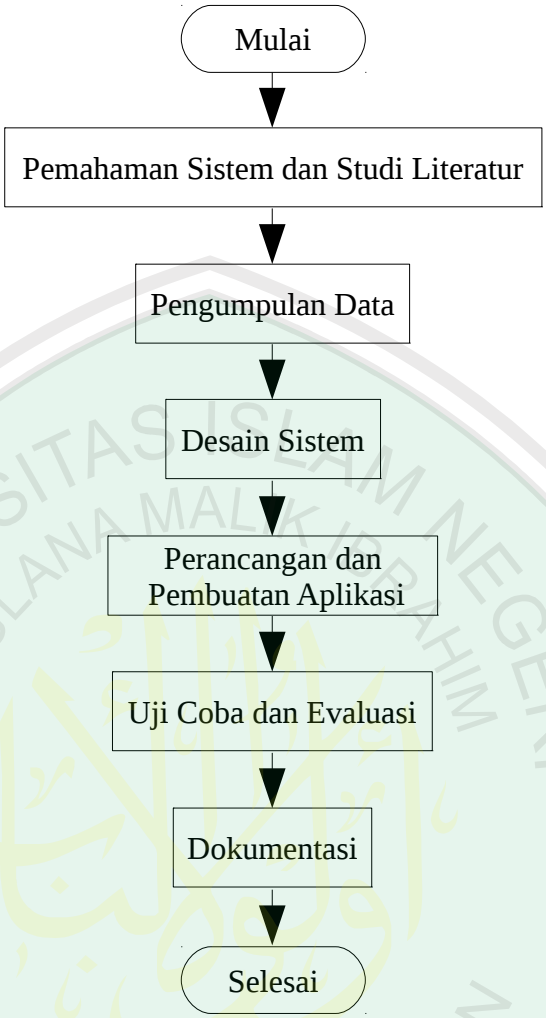
Penulis melakukan pengujian setiap kalimat dengan susunan yang telah tercantum di dalam berkas transcription, masing-masing diucapkan satu kali dengan variabel penelitian yang telah

dipaparkan di muka. Semua variabel tersebut dibuat seragam dengan nilai variabel yang digunakan dalam data suara acuan, sehingga banyaknya pengujian adalah 1 x 1 x 1 x 27 = 27 kali pengujian. Angka satu didapat dari banyaknya nilai yang terkandung dari sample rate, (yakni 16000), bit rate (yakni 16), dan banyaknya nilai tipe channel audio yang diinputkan dari mikrofon (yakni 1 untuk mono). Terakhir, angka 27 adalah jumlah kata yang akan diujikan. Dengan demikian, penulis akan melakukan pengujian total sebanyak 27 kali.

3.2.7. Dokumentasi

Pada poin ini, penulis melakukan dokumentasi hasil uji coba dan evaluasi setelah proses sebelumnya telah selesai dilalui. Dokumentasi tersebut dilakukan dalam penulisan laporan skripsi. Dengan demikian, hasil uji coba dan evaluasi dapat dijadikan pertimbangan untuk pengembangan dan penelitian lebih lanjut.

Secara ringkas, prosedur penelitian yang akan dilalui penulis dapat dijabarkan dalam gambar 3.3.



Gambar 3.8: Bagan alir prosedur penelitian

BAB IV

PEMBAHASAN

Pada bab ini, penulis memaparkan hasil implementasi beserta pembahasan tentang prosedur yang telah penulis ikuti. Secara garis besar, hal yang paling dominan untuk diulas adalah seberapa sukses implementasi sistem pengenalan ucapan yang telah penulis kembangkan, termasuk analisa kelebihan dan kekurangan sistem, beserta kendala yang dihadapi penulis di saat sistem tersebut sedang dikembangkan. Dengan demikian, penelitian ini diharapkan dapat menjadi bahan evaluasi untuk pengembangan penelitian lebih lanjut.

4.1. Hasil Implementasi

Secara terpisah, tidak banyak nilai variabel dalam berkas konfigurasi sphinx_train. Perubahan signifikan terletak pada penggunaan grammar pada saat proses training dan testing. Setelah melalui serangkaian data training, didapatkan nilai kesalahan kata rata-rata (*Word Error Rate/WER*) pada model tanpa grammar sebesar 60,7 % (atau 16 dari 28 kata yang dilatih dan diujikan) dan nilai kesalahan tingkat kalimat (*Sentence Error*) sebesar 100 %. (atau 5 dari 5 kalimat yang telah dilatih dan diujikan). Penggunaan grammar justru memperbesar nilai kesalahan tingkat kata menjadi 100 %. Persentase yang sama dengan kesalahan tingkat kalimat. Karena yang menjadi bahan penelitian di sini adalah pengujian kata, persentase kesalahan tingkat kalimat diabaikan. Dengan demikian, penulis memutuskan

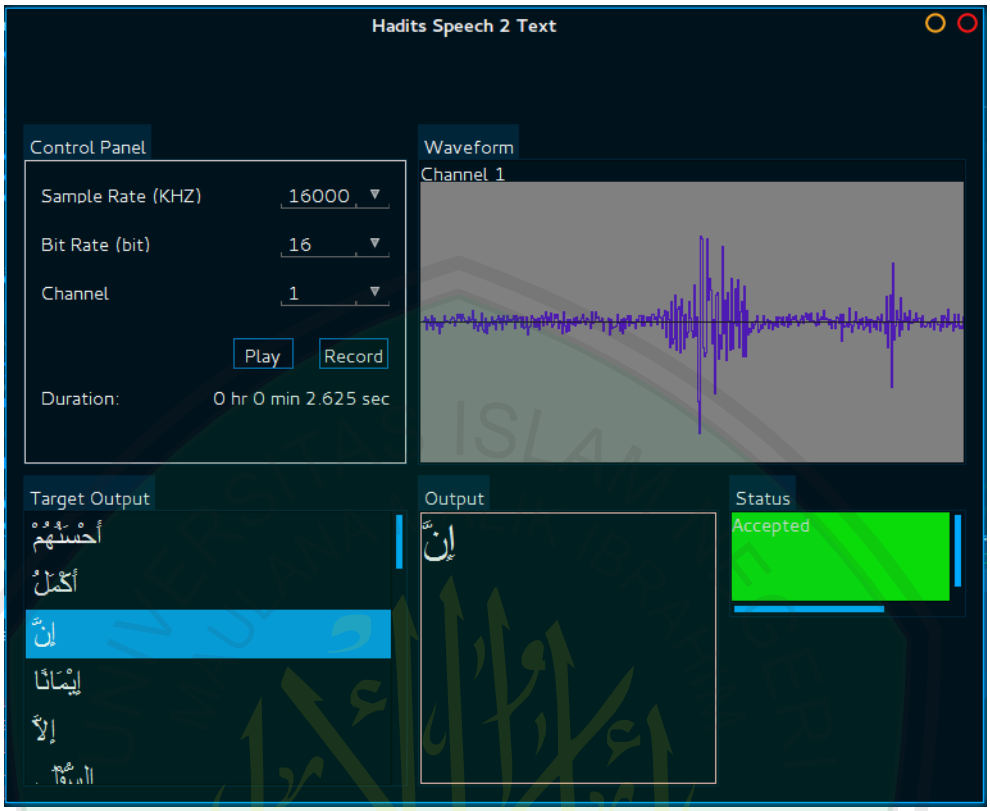
penggunaan model tanpa grammar untuk diujikan dalam program yang telah penulis desain. Hasil training dan testing lanjut dapat dilihat melalui file teks dalam subfolder *'result'*.

Sebagaimana telah dijelaskan di muka, penulis hanya menawarkan satu antarmuka yang memungkinkan pengaturan seluruh parameter yang dibutuhkan. Sebagaimana dijelaskan dalam subbab 3.2.4., penulis menampilkan berbagai variabel kontrol berupa sample rate, bit rate, dan channel yang telah diseragamkan dengan variabel yang dipakai dalam data suara acuan sebelum proses merekam suara, kemudian di sebelah kanan ada panel yang memuat bentuk gelombang setelah pengguna memilih berhenti merekam suara. Sedangkan panel berisi teks area yang tidak dapat diedit sebagai output yang memuat teks hasil pengenalan ucapan.

Setelah prosedur pembuatan grammar, language model, dictionary, dan Acoustic Model telah dilalui, penulis melakukan pengaturan dalam source code yang berfungsi menentukan berkas yang akan digunakan sebelum tahap implementasi. Berkas yang dimaksud adalah language model, dictionary, dan grammar. Khusus untuk grammar, penulis juga menentukan nama grammar yang akan dipakai dalam berkas grammar yang akan dijadikan acuan dalam menguji sistem pengenalan ucapan nanti. Sedangkan dalam model akustik, penulis menentukan folder yang memuat berbagai berkas hasil pelatihan model akustik. Berkas yang dimaksud telah dibahas pada subbab 3.2.3 ketika penulis menunjukkan langkah terakhir dalam

prosedur training model akustik. Dengan selesainya penentuan keempat komponen tersebut (language model, grammar, dictionary, dan acoustic modell), penulis siap melakukan pengujian pada antarmuka yang telah dirancang sebelumnya.

Penulis telah merancang bentuk antarmuka hasil implementari sistem pengenalan ucapan yang ditulis dalam bahasa pemrograman java. Mula-mula, penulis merekam suara melalui mikrofon. Selama proses merekam suara berlangsung, proses pengenalan berjalan seiring dengan berjalannya timer di bawah tombol, sehingga setiap suara yang keluar akan diproses oleh program. Untuk mengakhiri, penulis mengklik tombol stop, dan bentuk gelombang suara yang diinputkan keluar di sebelah kanan. Pada saat itulah timer berhenti dan menampilkan durasi suara yang telah diinputkan. Meski demikian, output belum keluar setelah proses merekam suara dihentikan. Berikutnya, penulis mengklik tombol play yang berperan dalam pembacaan data suara, sehingga dapat keluar dari speaker. Pada saat pembacaan data berlangsung, timer kembali berjalan dari waktu 0 detik hingga akhir data yang telah diinputkan Begitu proses pembacaan berakhir, output berupa teks keluar sebagai hasil representasi data suara yang telah diproses, dan timer akan berhenti, lalu kembali menampilkan durasi waktu data suara yang direkam.



Gambar 4.1: Bentuk Antarmuka hasil implementasi sistem pengenalan ucapan.

Langkah selanjutnya adalah dokumentsi hasil percobaan dalam bentuk tabel. Karena hanya ada 11 kata berhasil dikenali dalam proses training dan testing, penulis hanya melakukan percobaan 11 kali untuk kata yang dapat dikenali untuk memastikan ada tidaknya output yang muncul setelah suara diproses pasca rekaman. Data suara yang direkam, dengan durasi tergantung kata yang akan diucapkan pada saat pengujian.

Output Target	Output Pengenalan	Keterangan
أَحْسَنُهُمْ	-	diabaikan*
أَكْمَلُ	إِنَّ مِنْ	tidak sesuai
إِنَّ	إِنَّ	sesuai
إِيْمَانًا	-	diabaikan*
إِلَّا	-	diabaikan*
السُّفْلَى	-	diabaikan*
اللّٰهَ	-	tidak sesuai
الْجَمَالَ	-	diabaikan*
الْعُلْيَا	-	tidak sesuai
الْمُؤْمِنِيْنَ	مَنْ مِنْ	tidak sesuai
الْمُسْلِمُوْنَ	إِنَّ مِنْ إِنَّ	tidak sesuai
اَلْحَيَاءُ	-	diabaikan*
اَلْيَدُ	مِنْ إِنَّ	tidak sesuai
بِخَيْرٍ	-	diabaikan*
جَمِيْلٌ	اَلْيَدُ	tidak sesuai
خَيْرٌ	-	diabaikan*
خُلُقًا	-	diabaikan*
سَلِمَ	يَدِ	tidak sesuai
لِسَانِهِ	-	diabaikan*
مَنْ	مَنْ	sesuai
مِنْ	مَنْ	tidak sesuai
وَ	-	diabaikan*
يَأْتِيْ	-	diabaikan*
يَدِ	-	diabaikan*
يَدِهِ	-	diabaikan*
يُحِبُّ	-	diabaikan*
لَا	-	diabaikan*

UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM
PUSAT PERPUSTAKAAN

4.2. Pembahasan

Ketika membuka file Hadits_sample.align yang merupakan salah satu file hasil data training dan testing pengenalan ucapan sebelumnya, ada beberapa catatan yang akan diulas dalam poin ini. Kata yang gagal dikenali dengan benar akan ditandai dengan bintang (***). Kalimat pertama mengandung lima kata yang semuanya dikenali sebagai error setelah proses training dan testing. Artinya, tingkat akurasi kata dalam kalimat pertama adalah 0 %. Kalimat kedua mengandung tujuh kata, tiga di antaranya dianggap sebagai error. Dengan demikian, tingkat akurasi kata dalam kalimat kedua hanya 57,14 %. Kalimat ketiga terdiri atas 5 kata, dua di antaranya telah dikenali dengan benar, sehingga akurasi pelatihan dan pengujian pada kalimat ketiga mencapai 40 %. Kalimat keempat memiliki jumlah kata yang sama dengan kalimat ketiga. Namun, tiga dari lima kalimat yang telah disusun dengan benar, sehingga persentase akurasi kata dalam kalimat keempat adalah 60 %. Terakhir, kalimat kelima tersusun dari 6 kata. Dua di antaranya telah dikenali dengan benar, sehingga akurasi kata dalam kalimat kelima mencapai 33,33 %.

Dari total kata yang diuji (sebanyak 28 kata), sebelas di antaranya dapat dikenali dengan benar setelah pelatihan dan pengujian. Dengan demikian, akurasi kata untuk seluruh rangkaian kalimat yang diujikan sebanyak (11/28) * 100 % = 39,29 %. Penyebab terjadinya kesalahan dalam mengenali kata adalah banyaknya kata yang tidak berhasil dilatih dan diujikan, sehingga terhapuskan (*deletion*) pada saat training dan testing.

Ada satu kata yang diulang dalam proses pelatihan dan pengujian data, yakni مِنْ. Kata ini dapat dikenali pada kalimat kedua, namun dianggap sebagai error dalam kalimat kelima. Meski demikian, kata ini tetap akan penulis uji dalam program.

Langkah berikutnya adalah menggunakan model yang telah dilatih dan diujikan melalui *framework sphinxtrain*. Telah dijelaskan di muka bahwa hanya ada sebelas kata yang berhasil dikenali dengan baik melalui proses pelatihan. Dari sebelas kata tersebut, hanya dua kata yang sesuai dengan output target yang diharapkan, meski pengucapan harus dilakukan berulang-ulang, sebagaimana kata yang lain. Kata yang dimaksud adalah إِنَّ dan مَنْ. Keduanya merupakan kata yang sering muncul, bahkan dapat muncul berulang-ulang pada saat pengenalan ucapan yang lain. Sebagai contoh, ketika penulis hendak mengucapkan kata اَلْيَدُ, kedua kata tersebut dapat muncul secara berurutan, meski ada kemungkinan output yang keluar adalah kata yang lain dari keduanya. Dengan demikian, tingkat akurasi pengenalan kata pada saat pengujian lewat program adalah 2/27 * 100 % ~= 7,4 %. Hasil tersebut menunjukkan perlunya ada perbaikan lebih lanjut tentang model yang digunakan agar pengenalan dapat berjalan dengan lebih akurat.

4.3. Speech Recognition dalam Pelafalan Hadits Berbahasa Arab

Ketika ditanya tentang sumber rujukan baku sebagai pedoman dalam beragama Islam, sebagian besar kaum muslimin menjawab Kitab Allah berupa al-Qur'an, namun jarang disertai dengan Hadits Rasul Allah bernama

Muhammad. Kalau pun menjawab Hadits, mereka menempatkannya pada urutan kedua. Ini karena kebanyakan kaum muslimin lebih mengedepankan al-Qur'an sehingga Hadits Rasul Allah Muhammad cenderung dijadikan pelengkap dalam menjelaskan kandungan ayat al-Qur'an. Sejatinya, mereka juga sadar bahwa tidak semua persoalan dikupas lebih gamblang di dalam al-Qur'an, sehingga ketika tidak mendapati solusi berbagai persoalan di dalamnya, mereka merujuknya ke dalam Hadits Nabi, terlepas dari status Hadits yang dirujuknya. Hal ini merupakan cerminan dari perintah dan larangan yang disampaikan Rasul Allah dalam Surah al-Hasyr ayat 7

مَآ اَفَآءَ اللّٰهُ عَلٰى رَسُوْلِهٖ مِنْ اَهْلِ الْقُرٰى فَلِلّٰهِ وَ لِلرَّسُوْلِ وَ لِذِ الْقُرْبٰى وَ الْيَتٰمٰى وَ الْمَسٰكِيْنِ وَ ابْنِ السَّبِيْلِ كَيْلاَ يَكُوْنَ دُوْلَةً بَيْنَ الاَغْنِيَاءِ مِنْكُمْ . وَ مَا آتٰىكُمُ الرَّسُوْلُ فَخُذُوْا وَ مَا نَهٰىكُمْ عَنْهُ فَانْتَهُوْا . وَ اتَّقُوْا اللّٰهَ . اِنَّ اللّٰهَ شَدِيْدُ الْعِقَابِ (الحشر:7)

"Harta rampasan dari mereka yang telah Allah berikan kepada Rasul-Nya (yang berasal) dari penduduk beberapa negeri, adalah untuk Allah, Rasul-Nya, kerabat dekat, anak-anak yatim, orang-orang miskin, dan untuk orang-orang yang dalam perjalanan, agar (harta tersebut) tidak beredar pada orang kaya di antara kalian saja. Terimalah apa yang Rasul berikan kepada kalian, dan tinggalkanlah apa saja yang Rasulullah larang untuk kalian. Dan bertakwalah kepada Allah, sesungguhnya siksa Allah sangat keras." (QS. Al-Hasyr [59] : 7)

Dalam sebuah Hadits riwayat at-Tirmidzi dari Ibn Mas'ud, Rasul Allah Muhammad bersabda

نَضَّرَ اللّٰهُ امْرَءًا سَمِعَ مِنَّا شَيْئًا فَأَبْلَغَهُ كَمَا سَمِعَهُ فَرُبَّ مُبَلَّغٍ أَوْعَى مِنْ سَامِعٍ

Semoga Allah mencerahkan wajah seorang yang mendengar sebuah hadits dariku lalu dia menyampaikannya sebagaimana yang dia dengar. Maka terkadang orang yang menyampaikannya (Hadits Nabi, penj) lebih mampu menghafalnya untuk dirinya daripada hanya sekadar mendengar.

Hadits di atas merupakan keutamaan seseorang yang mempelajari hadits, menghafalnya, dan mengamalkannya. Sehingga mereka mendapat rahmat Allah atas upaya yang diperbuat ummat Nabi Muhammad dalam menjaga Hadits Nabi. Sebagaimana dalam kutipan hadits dari Ibn Abbas berikut

اللهُمَّ ارحَمْ خُلَفَائِيْ قُلْنَا يَا رَسُوْلُ الله, مَنْ خُلَفَاؤُكَ قَالَ الَّذِيْنَ يَرْوْنَ أَحَادِيْثِيْ وَ يُعَلِمُوْنَهَا النَّاسَ

Ya Allah, rahmatilah para penggantiku. Kami berkata, "Wahai Rasul Allah, siapakah para penggantimu ? " Rasul Allah Muhammad menjawab, "Orang-orang yang meriwayatkan Hadits dariku dan mengajarkannya kepada manusia" (HR. at-Thabrani dalam kitab al-Ausath)

Sedangkan di sisi lain, perkembangan ilmu pengenalan ucapan juga baru berkembang pada era ini, meski konsep berupa model matematika telah dikenal sejak puluhan tahun yang lalu. Kalau pun telah diimplementasikan dalam bahasa Arab, masih terbatas pada kasus tertentu, seperti al-Qur'an. Karena itulah, impelemntasi sistem pengenalan ucapan dalam pelafalan Hadits berbahasa Arab dianggap hal baru yang belum pernah dibahas dalam konsep pengolahan suara.

Atas dasar itulah, penulis mengharapkan adanya kontribusi yang signifikan dalam mengembangkan dan memperbaiki sistem pengenalan ucapan ini, khususnya berkaitan dengan Hadits Nabi, sehingga mendapat rahmat Allah dan pencerahan-Nya karena upaya penjagaan Hadits yang telah dilakukan oleh kaum muslimin seluruh dunia.

BAB V

PENUTUP

Bab ini merupakan ringkasan yang berisi pandangan penulis tentang hasil penelitian yang telah dilakukan sebagai bentuk representasi gambaran umum kinerja sistem pengenalan ucapan. Dengan kata lain, penulis hendak mengakhiri prosedur penulisan laporan terkait sistem tersebut dalam bab ini. Secara garis besar, penulis hanya membagi bab ini menjadi dua, yaitu kesimpulan dan saran.

4.4. Kesimpulan

Prosedur penelitian yang telah ditempuh dan didokumentasikan dalam laporan ini, telah menghasilkan kesimpulan sebagai berikut.

1. Dari 28 kata yang diujikan melalui proses training dan testing, sebelas di antaranya dikenal dengan benar. Sehingga akurasi yang dihasilkan adalah 39.29 %
2. Berdasarkan hasil pengujian melalui program yang telah dibuat, dua kata telah dikenali dengan benar, dari 27 kata yang hendak diujikan sehingga menghasilkan akurasi sebesar 7,4 %

4.5. Saran

Kesimpulan yang telah penulis utarakan sebelumnya mendorong penulis memberikan saran sebagai berikut

1. Isi data penunjang (Language Model, Grammar, Dictionary, dan Acoustic Model) sebaiknya diperbaiki dan diperkaya. Terutama dalam pemodelan Akustik, terkait data suara yang menjadi acuan dalam proses pemodelan. Sebab, dengan kayanya data penunjang yang diberikan, proses pengenalan diharapkan menjadi lebih akurat.
2. Perlu adanya pengujian model akustik tersendiri untuk memastikan bahwa model akustik yang telah dirancang dapat siap digunakan. Itu pun harus sesuai dengan data yang disiapkan menjelang proses pemodelan akustik

DAFTAR PUSTAKA

Animasahun, I. O. and Popoola, J. J. *Application of Mel Frequency Ceptrum Coefficients and Dynamic Time Warping For Developing an Isolated Speech Recognition System.* International Journal of Science and Technology Volume 4 No. 1, January, 2015. http://www.journalofsciences-technology.org/archive/2015/jan_vol_4_no_1/8916691415784643.pdf (diakses tanggal 28 April 2015 pukul 10.06 WIB)

Gupta, Shikha; Jaafar, Jafreezal; wan Ahmad,Wan Fatimah; dan Bansal, Arpit. *Feature Extraction Using MFCC.* Signal & Image Processing: An International Journal (SIPIJ) Vol.4, No.4, August 2013 http://aircconline.com/sipij/V4N4/4413sipij08.pdf (diakses tanggal 28 April 2015)

Hunt, Andrew. 2000. *JSpeech Grammar Format.* Sun Microsystems, Inc. https://www.w3.org/TR/jsgf/ (diakses tanggal 20 Mei 2016 pukul 04.45 WIB)

Ir. P. Wiggers; Dr. drs. L.J.M. Rothkrantz. 2003. *Automatic Speech Recognition using Hidden Markov* Models. Data and Knowledge Systems Group http://www.kbs.twi.tudelft.nl/docs/syllabi/speech.pdf (diakses tanggal 20 Mei 2015 pukul 12.15 WIB)

Lajnah Pentashihan Mushaf Al-Qur'an. 2012. *Tafsir Alqur'an Tematik: Hukum, Keadilan, dan Hak Asasi Manusia.* Jakarta: Aku Bisa.

Meseguer, Noelia Alcaraz. 2009. *Speech Analysis for Automatic Speech Recognition.* Norwegian University of Science and Technology Department of Electronics and Telecommunications. http://www.diva-

portal.org/smash/get/diva2:347957/FULLTEXT01.pdf (diakses tanggal 25 November 2015 pukul 12.45 WIB)

Picone, Dr. Joseph. 1996. *Fundamentals of Speech Recognition: A Short Course.* Institute For Signal And Information Processing. Institute for Signal and Information, Processing Department of Electrical and Computer Engineering Mississippi State University http://www.stillhq.com/diary/asr-short.pdf (diakses tanggal 17 Februari 2015)

Rabiner, Lawrence R. 1989. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, Vol. 77, No. 2, February 1989 http://www.eecis.udel.edu/~shatkay/Course/papers/RabinerHMMTutorial.pdf (Diakses tanggal 20 Mei 2015 pukul 12.39 WIB)

Razak, Zaidi; Ibrahim, Noor Jamaliah; Idris, Mohd Yamani Idna; Tamil, Emran Mohd;Yakub, Mohd; Yusoff, Zulkifli Mohd; Rahman, Noor Naemah Abdul. 2008 Quranic Verse Recitation Recognition Module for Support in j-QAF Learning: A Review. IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.8, August 2008 http://paper.ijcsns.org/07_book/200808/20080831.pdf (diakses tanggal 17 Februari 2015)

Renals, Steve. 1996. *Speech Recognition*. http://dsp-book.narod.ru/rec-notes.pdf (diakses tanggal 17 Februari 2015)

Sakoe, H., & Chiba, S. 1978. *Dynamic Programming Algorithm Optimization for Spoken Word Recognition.* Acoustics, Speech and Signal Processing, IEEE Transactions on, 26 (1), 43-49.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.114.3782&rep=rep1&type=pdf (diakses tanggal 22 Mei 2015)

Stamp, Mark. *A Revealing Introduction to Hidden Markov Models*. Department of Computer Science, San Jose State University. September 2012 https://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf (diakses tanggal 21 Mei 2015 pukul 12.45 WIB)

Sugiyono, Prof. Dr. 2008. *Metode Penelitian Kuantitatif, Kualitatif, dan R&D*. Bandung: ALFABETA

Walker, Willie; Lamere, Paul; Kwok, Philip; Raj, Bhiksha; Singh, Rita; Gouvea, Evandro; Wolf, Peter; dan Woelfel, Joe. 2004. *Sphinx-4: A Flexible Open Source Framework for Speech Recognition.* Sun Microsystems Inc.

Yekache, Yacine; Mekelleche, Yekhlef;Kouninef, Belkacem. 2012. *Towards Quranic Reader Controlled by Speech.* Institut National des Télécommunications et des TIC, ALGERIA http://arxiv.org/pdf/1204.1566.pdf (diakses tanggal 6 Oktober 2014 pukul 10.19 WIB)

Yusuf, Ahmad Muhammad. 2009. *Ensiklopedi Tematik Ayat Al-Qur'an dan Hadits*. Jakarta: Widya Cahya

underflow: definition of underflow in oxford dictionary (American English). http://www.oxforddictionaries.com/definition/american_english/underflow [diakses tanggal 15 April 2015 Pukul 12.30]

Mel Frequency Cepstral Coefficient (MFCC) tutorial – Practical Cryptography http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ (Diakses tanggal 26 April 2015 pukul 22.47 WIB)

Building Language Model [CMUSphinx wiki] http://cmusphinx.sourceforge.net/wiki/tutoriallm (diakses tanggal 12 April 2016 Pukul 10.13 WIB)

Training Acoustic Model [CMUSphinx wiki] http://cmusphinx.sourceforge.net/wiki/tutorialam (diakses tanggal 14 April 2016 pukul 09.12 WIB)