APFOS: PERANGKAT LUNAK PENGISIAN FORM APLIKASI TERTULIS DENGAN MENGGUNAKAN PENGENALAN SUARA

Iping Supriana Suwardi[1] dan Reza Adicipta[2]
Program Studi Informatika, Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132
E-mail: iping@informatika.org[1], if11059@students.if.itb.ac.id[2]

ABSTRAKSI

Salah satu aplikasi dari sistem pengenalan suara adalah sebagai data entry. Pada tulisan ini dijelaskan tentang pembangunan aplikasi perangkat lunak data entry form aplikasi tertulis dengan menggunakan sistem pengenalan suara. Sistem pengenalan suara yang digunakan adalah Sphinx-4 yang berbasis hidden Markov model. Agar perangkat lunak dapat digunakan untuk berbagai form aplikasi, Penulis merumuskan suatu format form aplikasi dalam XML. Pada bagian akhir dari tulisan ini akan diberikan diskusi dan hasil pengujian perangkat lunak.

***Kata kunci**: data entry, speech recognition, sphinx-4*

1. PENDAHULUAN

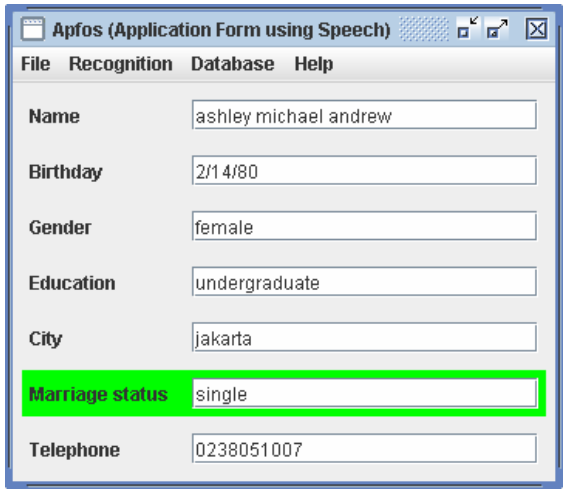
Pengenalan suara adalah kemampuan untuk menerima masukan berupa suara dan menghasilkan transkrip tekstual dari suara tersebut sebagai keluaran[1]. Salah satu area aplikasi dari pengenalan suara adalah sebagai alternatif antarmuka untuk *data entry* ketika tangan dan mata tidak bisa digunakan atau sibuk. Salah satu contoh aplikasi yang dapat dikembangkan adalah penerapan pengenalan suara pada pengisian form aplikasi tertulis[2]. Sistem menerima masukan berupa jawaban dari pengisi form dalam bentuk suara dan merubahnya menjadi teks lalu menyimpannya. Metode ini diharapkan dapat memberikan efisiensi lebih dibandingkan metode konvensional.

Masukan dari pertanyaan pada form aplikasi tertulis pada dasarnya dapat dipenuhi hanya dengan elemen *textbox*. Selain *textbox*, ada juga *checkbox* dan *radiobutton* yang dapat memberikan *constraint*. Berdasarkan survey[3], *textbox*, *checkbox*, dan *radiobutton* adalah elemen yang sering digunakan pada form aplikasi tertulis. Untuk selanjutnya pertanyaan-pertanyaan pada form aplikasi akan dinamakan *elemen form*. Dengan demikian suatu form aplikasi terdiri dari suatu set elemen form.

2. APPLICATION FORM USING SPEECH (APFOS)

Penulis mengembangkan sebuah aplikasi perangkat lunak untuk mencoba menjawab tantangan ini. Perangkat lunak tersebut dinamakan *Application Form using Speech* (APFOS) yang dibangun dengan bahasa pemrograman Java™ (Gambar 1). Untuk sistem pengenalan suara, penulis menggunakan Sphinx-4 yang merupakan *open platform* untuk riset di bidang pengenalan suara berbasis *hidden Markov model*[4]. Data pengisian disimpan dalam sebuah DBMS dan dapat dieksport kedalam format XML dan CSV.

APFOS dapat dipakai untuk berbagai form aplikasi. Spesifikasi mengenai form aplikasi yang ingin digunakan dijabarkan dalam arsip spesifikasi berbasis XML yang berisi deskripsi elemen-elemen form dan konektivitas basis data.

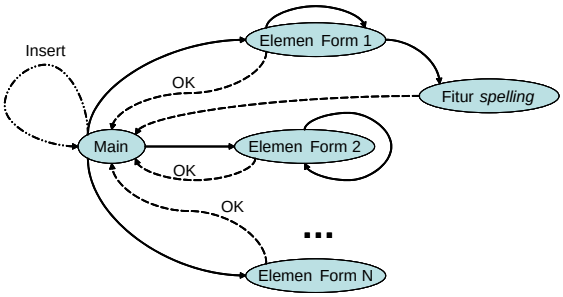


Gambar 1. *Application Form using Speech* (APFOS)

3. DIAGRAM PERINTAH

Pada bagian ini akan dijelaskan mengenai diagram tingkatan perintah masukan yang dimiliki APFOS (Gambar 2). Pada tingkatan paling atas (*main*), APFOS menerima perintah "*insert*" untuk memasukkan data form ke DBMS dan perintah-perintah untuk berpindah ke masing-masing elemen form untuk pengisian. Perintah perpindahan elemen form dijabarkan didalam arsip konfigurasi form. Untuk kembali ke tingkat yang lebih atas, perintah yang digunakan adalah "*OK*".

Pada tingkat elemen form, APFOS menerima perintah yang didefinisikan sendiri melalui arsip tata bahasa elemen form. Tata bahasa yang digunakan adalah *context-free grammar* yang dinyatakan dengan *Java Speech Grammar Format* (JSGF)[5]. Sebagai contoh adalah elemen form *gender*, elemen form ini mempunyai tata bahasa sebagai berikut: `public <gender> = male | female;`



Gambar 2. Diagram perintah APFOS

Jika elemen form memerlukan masukan yang tidak dapat dibaca, seperti kode mata kuliah, atau masukan yang berpotensi memberikan ambiguitas seperti nama orang, maka elemen form dapat diberikan pilihan untuk pengejaan masukan pada arsip spesifikasi form aplikasi. Karakter alfanumerik dieja dengan menggunakan kaidah *NATO phonetic alphabet*[6] untuk menghindari ambiguitas.

4. FORMAT XML FORM APLIKASI

Format form aplikasi mempunyai dua fungsi utama, yaitu:

1. mendeskripsikan elemen form yang diinginkan dalam suatu form.
2. menentukan konfigurasi basisdata yang digunakan untuk penyimpanan data.

```
<form name=[form name]>
  <database url=[database location]>
    <username>[username]</username>
  </database>

  <element name=[element name]
      grammar_file=[grammar file]
      column_name=[column name]
      type=[type]
      spell=[none|numeric|alphanumeric]>
    <hint>[hint]</hint>
  </element>
</form>
```

Contoh 1. Spesifikasi form

Contoh 1 adalah spesifikasi arsip. *Root tag* dari arsip adalah `<form>`. `<form>` menerima satu atribut, yaitu name, yang mendefinisikan nama form.

Tag `<database>` mendeskripsikan koneksi basisdata yang diperlukan untuk melakukan operasi *insert* terhadap data pengisian. *Tag* ini hanya boleh terdapat satu kali dalam arsip. Atribut yang diterima oleh tag ini adalah URL yang mendeskripsikan lokasi basisdata. *Subtag* <username> menjelaskan *username* untuk basisdata. *Password* tidak dituliskan untuk menjaga keamanan, tetapi akan ditanyakan oleh aplikasi saat berjalan.

Tag `<element>` mendeskripsikan elemen-elemen form. *Tag* ini bisa terdapat lebih dari satu. Atribut yang ada, adalah:

1. `name`
   Menderskripsikan nama elemen form dan merupakan perintah untuk masuk ke pengisian elemen form ybs.
2. `grammar_file`
   Masing-masing elemen form mempunyai arsip tata bahasanya sendiri yang dinyatakan dengan *Java Speech Grammar Format*[5].
3. `column_name`
   Mendeskripsikan nama kolom di dalam basis data yang akan diisi dengan informasi pengisian di elemen form tersebut.
4. `type`
   Menjelaskan tipe dari elemen form di dalam basis data, seperti: *string*, *integer*.
5. `spell`
   Untuk input yang rumit, seperti kode mata kuliah yang tidak bisa diucapkan, APFOS menyediakan fitur pengejaan. Terdapat tiga pilihan isian untuk atribut ini, yaitu: *none*, *numeric*, dan *alphanumeric*. *None* berarti fitur *spelling* dimatikan. *Numeric* berarti *spelling* di-set untuk angka, misalnya untuk nomor telepon. *Alphanumeric* berarti *spelling* di-set untuk angka dan alphabet, misalnya untuk kode mata kuliah IF101.

```
<form name="Gender Form">
  <database url="jdbc://localhost">
    <username>adicipta</username>
  </database>
  <element name="Gender"
           grammar_file="gender"
           column_name="gender"
           type="string"
           spell="none">
    <hint>Please insert your gender info.
          Say 'male' or 'female'</hint>
  </element>
</form>
```

Contoh 2. Form dengan satu elemen form

Selain itu, terdapat *subtag* `<hint>` yang mendeskripsikan petunjuk cara pengisian, seperti apa yang harus diucapkan untuk mengisi elemen form ini. Petunjuk ini ditampilkan saat pengguna masuk ke elemen form ybs.

Contoh 2 merupakan arsip form *'Gender Form'* yang mempunyai satu elemen form berupa *'gender'*.

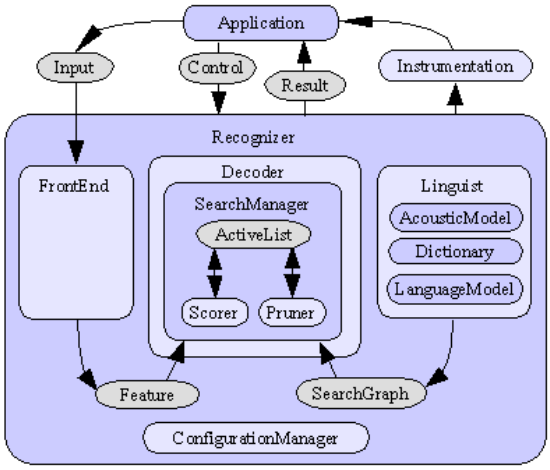
5. SPHINX-4

Sphinx merupakan sistem pengenalan suara berbasis *hidden Markov model* yang dikembangkan oleh *Carnegie Mellon University* yang berkolaborasi dengan *Sun Microsystems* dan *Mitsubishi Electric Research Lab*. Sphinx terdiri dari versi 1, 2, 3, dan 4. Sphinx 1, 2, dan 3 ditulis dalam bahasa c, sedangkan sphinx-4 ditulis dalam bahasa Java™ untuk menjamin interoperabilitas.

Sphinx-4 tersusun atas modul-modul yang dapat diganti secara fleksibel pada saat *run-time*. Pada gambar. 3 terdapat arsitektur dari Sphinx-4. Terdapat tiga modul utama pada Sphinx-4, yaitu: *FrontEnd*, *Decoder*, dan *Linguist*. *FrontEnd* menerima masukan sinyal suara dan menghasilkan serangkaian *Feature*. *Linguist* membangun *SearchGraph* dari dari *AcousticModel*, *LanguageModel*, dan *Dictionary*. *SearchManager* yang terdapat di dalam modul *Decoder* akan menggunakan *SeachGraph* dan *Feature* untuk melakukan pencarian yang akan menghasilkan *Result*. Selain modul-modul diatas, juga terdapat *ConfigurationManager* yang bertugas untuk

 ISSN: 1907-5022

menyusun konfigurasi modul-modul Sphinx-4. Konfigurasi modul-modul Sphinx-4 dijabarkan pada sebuah arsip konfigurasi berbasis XML.



Gambar 3. Arsitektur Sphinx-4

6. IMPLEMENTASI

Pada bagian ini akan dijelaskan mengenai implementasi Sphinx-4 pada APFOS.

6.1 Ekstraksi Fitur

Untuk mengekstrak fitur, penulis menggunakan teknik-teknik pemrosesan sinyal. Objek pemrosesan sinyal yang digunakan adalah:

1. `Microphone`
   Menangkap data audio dari sistem dan merubahnya menjadi objek Data.
2. `SpeechClassifier`
   Mengklasifikasikan audio menjadi *speech* dan *non-speech*.
3. `SpeechMarker`
   Memberikan label SPEECH_START dan SPEECH_END pada awal dan akhir dari audio *speech*.
4. `NonSpeechDataFilter`
   Membuang *non-speech* audio, yaitu yang berada diantara SPEECH_END dan SPEECH_START.
5. `Preemphasizer`
   Sinyal suara umumnya mengalami pelemahan, *Preemphasizer* meningkatkan magnitud dari komponen frekuensi tinggi agar dapat dianalisis secara lebih baik oleh filter[7].
6. `RaisedCosineWindower`
   Membagi objek Data menjadi frame-frame yang overlap dengan frame tetangganya. Ini untuk menghindari kehilangan informasi pada area pergantian frame
7. `DiscreteFourierTransform`
   Menghitung transformasi Fourier diskrit dari frame masukan dengan menggunakan *Fast Fourier Transform*.
8. `MelFrequencyFilterBank`
   Memfilter input spektrum melalui sejumlah *mel-filter* untuk menghasilkan *mel-spectrum*.
9. `DiscreteCosineTransform`
   Mengaplikasikan Transformasi Kosinus Diskrit pada mel-spectrum untuk menghasilkan *Mel Frequency Cepstral Coefficient* (MFCC)[8].
10. `LiveCepstralMeanNormalization`
    Menghitung CMN dari data cepstral. Teknik ini untuk mengurangi distorsi pada channel transmisi. Data keluaran adalah data cepstral yang ternormalisasi.
11. `DeltasFeatureExtraction`
    Menghitung delta dan double delta dari input cepstrum. Hasil keluaran adalah vektor fitur yang akan digunakan oleh *Decoder*.

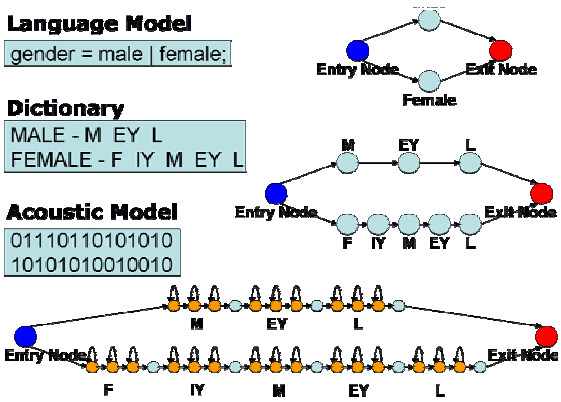
6.2 *Linguist*

Telah dijelaskan sebelumnya bahwa *Linguist* membangkitkan *SearchGraph* yang merupakan ruang pencarian yang akan digunakan oleh *Decoder*. *Linguist* menggunakan basis pengetahuan yang terdiri dari tiga bagian, yaitu: *AcousticModel*, *LanguageModel*, dan *Dictionary*. APFOS menggunakan kelas FlatLinguist sebagai *Linguist*, karena implementasi tersebut cocok untuk model bahasa berbasis *context-free grammar*[4].

Pembangkitan graf pencarian dapat dilihat pada gambar 4. Pertama-tama graf dibangun dari Language Model, kemudian setiap kata didalam language model akan ditranslasikan ke deretan unit suara dengan melihat Dictionary. Yang terakhir adalah mengganti setiap node unit suara menjadi HMM-nya dengan melihat model unit suara didalam Acoustic Model.

6.2.1 *Acoustic Model* / Model Akustik

AcousticModel berisi model HMM untuk tiap unit suara. Unit-unit suara yang dipakai mengikuti kaidah ARPAbet[9]. Model HMM dilatih menggunakan algoritma *forward-backward*[10] dengan *corpus speech* Wall Street Journal[11]. Model dilatih dengan menggunakan Sphinx Trainer[4].



Gambar 4. Contoh Pembangkitan SearchGraph untuk mengenal kata "female" atau "male"

6.2.2 *Language Model* / Model Bahasa

LanguageModel menggunakan tata bahasa *context-free grammar*(CFG). Implementasi di Sphinx menggunakan kelas JSGFGrammar yang mendukung *Java Speech Grammar Format* (JSGF)[5].

Setiap elemen form diharuskan mendeskripsikan model bahasanya sendiri, contohnya: elemen form *gender* mempunyai model bahasa seperti yang telah dijelaskan sebelumnya.

6.2.3 *Dictionary* / Kamus

Dictionary berisi data pengucapan kata-kata yang digunakan pada model bahasa. Kata-kata ditranslasikan menjadi susunan unit-unit suara yang dipakai di model akustik. Unit-unit suara mengikuti kaidah ARPAbet. *Dictionary* yang dipakai pada APFOS adalah *CMU Pronouncing Dictionary*[12] yang berisi ±130.000 pengucapan kata-kata dalam bahasa Inggris. Berikut adalah kata "*speech*" dan "*recognition*" dari dictionary.

```
SPEECH          S P IY CH
RECOGNITION     R EH K AH G N IH SH AH N
```

6.3 Decoder

Decoder terdiri dari *SearchManager* yang bertugas melaksanakan strategi pencarian. *SearchManager* menggunakan algoritma *token passing*[13] ketika menyusuri *SearchGraph*. Objek Token yang masih layak untuk dijelajahi disimpan dalam kelas ActiveList. Kelas *SearchManager* yang digunakan dalam APFOS adalah SimpleBreadthFirstSearchManager, sesuai namanya, *SearchManager* ini menggunakan *Breadth First Search* sebagai algoritma pencariannya.

Untuk mengetahui nilai probabilitas dari Feature terhadap suatu model HMM, digunakan objek Scorer. Objek Scorer menggunakan algoritma *Viterbi*[14]. Untuk menjaga agar ukuran ActiveList tetap kecil sehingga pencarian menjadi lebih cepat, digunakan objek Pruner untuk menghapus *Token* yang nilai probabilitasnya dibawah ambang batas.

7. PENGUJIAN

Sebagai pengujian, penulis membuat sebuah form tes dengan elemen form sebagai berikut:

1. Nama
2. Tanggal Lahir
3. Jenis Kelamin
4. Pendidikan
5. Kota
6. Status Pernikahan
7. Telepon

Penulis menggunakan APFOS untuk mengisi form dan mengukur waktunya. Sebagai pembanding, penulis menggunakan form yang sama untuk pengisian manual. Tabel 1 menunjukkan perbandingan hasil yang didapat.

Dari hasil pengujian, APFOS masih mengalami kelemahan jika dibandingkan dengan metode manual dalam hal kecepatan, tetapi APFOS telah berhasil untuk mengeliminasi penggunaan *keyboard* dalam pengisian dan dengan hasil pengisian yang benar.

Tabel 1. Hasil Pengujian

Jumlah Pengisian	*Waktu APFOS*	*Waktu Manual*
20	37	22
Rata-rata	1.85 menit	1.1 menit

Penyebab dari lamanya pengisian form dengan APFOS adalah kesalahan yang ditimbulkan dari pengenalan suara akibat *noise* dari lingkungan. Oleh karena itu diperlukan pengulangan dan pengkoreksian isian yang salah. APFOS sendiri telah dikembangkan dengan sistem pengisian yang memungkinkan pengguna untuk mengoreksi isiannya.

Tingkat persentase ketepatan sistem APFOS dalam mengenali suatu kata sudah tergolong tinggi, tetapi dengan semakin banyaknya elemen form yang harus di-input, maka persentase ketepatan sistem dalam mengenali seluruh isian akan menurun, sehingga besar kemungkinan pengguna harus mengoreksi beberapa isiannya pada form yang panjang.

8. RENCANA MASA DEPAN

Untuk saat ini, APFOS hanya dapat digunakan untuk form yang menggunakan bahasa Inggris. Rencana ke depan adalah melatih basis pengetahuan APFOS agar dapat digunakan untuk form berbahasa Indonesia. Oleh karena itu dibutuhkan *corpus speech* dan arsip *dictionary* berbahasa Indonesia.

Antarmuka aplikasi dapat ditingkatkan dengan fitur *text-to-speech* sehingga pengguna tidak perlu menggunakan indra penglihatannya untuk melihat kebenaran masukan dan dapat melakukan dialog dengan aplikasi.

DAFTAR PUSTAKA

[1] Rodman, Robert D., (1999). *Computer Speech Technology*. Artech House.
[2] Klevans, Richard L. Rodman, Robert D (1997). *Voice Recognition*. Artech House.
[3] Adicipta, Reza (2005). *Survey Form Aplikasi Tertulis*. Bandung.
[4] Walker, W. Singh, R. Raj, B (2004). *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*. Sun Microsystems.
[5] *Java speech API grammar format* (JSGF). http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/
[6] International Telecommunication Union (1959), *Radio Regulations*. Geneva.
[7] Rabiner, L. Juang, BH. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.
[8] Deller, J. (2000). *Discrete-Time Processing of Speech Signal*. MacMillan Publishing.
[9] Jurafsky, Daniel. Martin, James H (2000). *Speech and Language Processing*. Prentice Hall.
[10] Huang, XD. (1993). *Large-Vocabulary Speaker-Independent Continuous Sppech Recognition with Semi-Continuous Hidden Markov Model*. Carnegie Mellon University
[11] *Linguistic Data Consortium*. CSR-1(WSJ0) complete. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S6A
[12] Carnegie Mellon University. *CMU pronouncing dictionary*. http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[13] S. J. Young, N. H. Russell, J. H. S. Russell (1989). *Token passing: A simple conceptual model for connected speech recognition systems*. Cambridge University Engineering Dept, UK.
[14] Huang, XD. Acero, (2001). *A Spoken Language Processing – A Guide to Theory, Algorithm, and System Development*. Prentice-Hall.