Vol.2011-SLP-89 No.28
2011/12/20

# 音声対話システムにおいて複数の言語理解モデルの利用を容易にするツールキット：MLUTK

竹内誉羽[†1]　中野幹生[†1]
森祥二郎[†2]　駒谷和範[†2]

音声対話システムでは言語理解部が重要であり，複数の言語モデルと複数の言語理解モデルを用いることにより，音声理解の精度を高める手法が提案されている．我々は複数の言語理解モデルをオンラインで動作する音声対話システムに比較的容易に組み込むためのツールキットを作成した．このツールキットはオブジェクト指向言語により実装されており，ユーザが独自の言語理解モデルクラスを実装して組み込めるように設計されている．また，複数の言語理解モデルの利用を容易にするために音声認識文法や学習のための訓練データの書式フォーマットも設定した．

# A Toolkit that Facilitates Multiple Language Understanding Models in Spoken Dialog Systems: MLUTK

Johane Takeuchi,[†1]　Mikio Nakano,[†1]
Shojiro Mori[†2] and　Kazunori Komatani[†2]

Language understanding parts are generally crucial for spoken dialog systems. It was suggested that exploiting several language models and multiple language understanding led to improving accuracies of understanding user's utterances in spoken dialog systems. We created a toolkit that facilitates multiple language understanding models for online spoken dialog systems. Our toolkit is implemented with a object-oriented programming language so that the users can program their own language understanding models. We also support formats like a speech recognition grammar, which ease the actual usage of multiple language understanding models with this toolkit.

| | |
|---|---|
| **ユーザ発話：** | **午後 1 時から午後 2 時までです.** |
| **音声認識結果：** | “午後 1 時から午後 2 時まで” |
| **音声理解結果：** | “start-time:13 end-time:14 type:video-record” |

**図 1**　ビデオの録画予約における音声認識結果と音声理解結果の例

## 1. はじめに

音声対話システムではユーザの発話から得られた意味表現に基づいて応答を生成するため，発話を意味表現に変換する音声理解部が重要である[1)]．音声理解は，音声をテキストに変換する音声認識と，得られたテキストから意味表現に変換する言語理解の二つのプロセスからなる（**図 1**）．音声認識には音響モデルと言語モデルが必要であるが，音響モデルは通常，音声対話システムのタスクドメインに依存しないので，ここでは言語モデルと言語理解モデルのみを対象とする．

さて，これらの言語モデルと言語理解モデルを複数使用すると，多様な発話に対して高精度な音声理解を実現できることが勝丸等により報告されている[1)]．本報告でも勝丸等の呼称を使用して，この複数の言語モデルと言語理解モデルを使う方法を MLMU(Multiple Language models and Multiple Understanding models) と呼ぶことにする．最近の音声認識器の多くは複数の代表的な言語モデル（例えば，文法モデルと N-gram モデル）を容易に使えるようになっている．これに対して，複数の言語理解モデルを使って意味理解を行うためには，スクラッチから自分で実装するしかない．そこで我々は複数の言語理解モデルを使った意味理解を実現するためのソフトウェア・ツールキット，MLUTK(Multiple Language Understanding Tool-Kit) を作成した．この MLUTK は任意の音声対話システムに組み込んで使うことができる．

## 2. MLMU と関連研究

MLMU では，複数の言語モデル・言語理解モデルを用いて両者のあらゆる組み合わせによる音声理解を行う（**図 2**）．その後，それらの組み合わせによる音声理解結果から，いず

†1 (株) ホンダ・リサーチ・インスティチュート・ジャパン
Honda Research Institute Japan Co., Ltd.
†2 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University

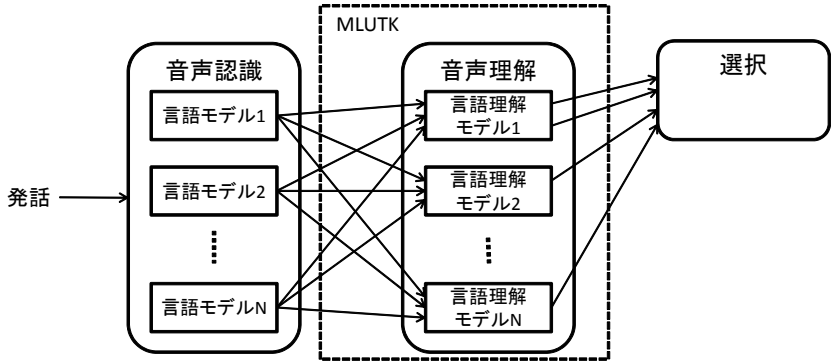


**図 2** MLMU における音声理解の流れと MLUTK の対象範囲

れかを選択する[1]．複数の言語モデルを使えば，音声認識結果に正解のキーワードを含む確率が上がるため，言語理解精度の向上に貢献する．さらに勝丸等による別の報告[2] によれば，統計学習によるモデルを含む複数の言語理解モデルを使う方が単一の言語理解モデルを使うより理解精度が高い．また Raymond 等の報告[3] によれば，対話ドメインによって理解精度がもっとも高い言語理解モデルが異なる．従ってマルチドメイン対話システムで高い理解精度を得るためには，複数の言語モデルを使うだけでなく，複数の言語理解モデルを使用したほうが良いと強く予想されるのである．ただしマルチドメイン対話システムにおける具体的な評価は今後の課題である．マルチドメイン対話システムを構築する際に，すべてのドメインで等しく言語理解のための学習データを用意することは，大変な労力を要する作業であり，学習を必要としない言語理解モデルも合わせて使用できる方が実用的でもある．

MLUTK は我々が開発してきたロボット向けマルチドメイン音声対話システム構築ツールである RIME-TK[4] と深く関連しており，MLUTK は RIME-TK に組み込んで使用できる．しかしながら MLUTK 自身は RIME-TK に依存しないように作成した．その結果，RIME-TK 以外の音声対話ツールと組み合わせて使用することも可能である．また，RIME-TK を MLUTK に対応できるように一部変更を加えたが，これについては本報告ではこれ以上触れない．

その他の関連研究としては，音声対話インターフェースを作成するためのツールキット[5),6)] が挙げられるだろう．これらに対して MLUTK は，音声対話における MLMU を比較的容易に実装可能とすることで高精度な音声理解を実現するためのツールキットである．

## 3. MLUTK

MLUTK では，MLMU における一連の処理のうち複数の言語理解モデルを管理して，複数の音声理解結果を出力する部分までを主な対象範囲とする（図 2）．音声認識器そのものは，市販あるいはオープンソースの自動音声認識ソフトで複数の言語モデルを使用できるものを使うことを想定している．ただし音声理解に必要な認識文法などは音声認識器と密接に関連しているため，それらのフォーマットの変換ツールなども整備している．また音声理解結果の選択について勝丸等[1] は，CMBS(Confidence-Measure-Based Selection) というロジスティック回帰に基づく手法も提案しているが，今のところ MLUTK としてはその実装を提供していない．どのような選択手法を用いるかは言語理解モデルの実装にもよるので，開発者が最適なものを実装するのが望ましいと考える．

### 3.1 目　　標

我々は MLUTK を作成するにあたって，以下の目標を掲げた．

(1) ユーザが自分で言語理解モデルを実装して拡張できるようにする．
(2) 複数の言語理解モデルを使用する際の管理をできる限り容易にする．

目標 (1) については，言語理解モデルを自由に入れ替えられるようになっていて，かつ後から言語理解モデルを作成して追加できるようにすることである．これによって，ユーザが独自の言語理解モデルを組み込んで評価したり実用化したりすることが可能になる．目標 (2) は，実際に複数の言語理解モデルを使う際に必要となる事柄である．最近では，言語理解のために意味タグのついた音声認識文法記述が使われ，これを記述すれば音声認識器によっては認識結果と同時に言語理解結果も出力できるものもある[7]．しかしながら，音声認識器に組み込まれていない言語理解モデルも合わせて使う場合，困難につきあたる．例えば CRF（条件付き確率場：Conditional Random Field）による系列ラベリングの手法を使った言語理解モデル[3] は， 単語ごとにタグ付して学習させなければならないが，この “学習のためのタグ情報” と “意味タグ付きの文法” を別個に「管理」するのは，スケールが大きくなるに従って煩雑になりすぎる．従って，このような管理を出来うる限り容易にすることが求められる．

### 3.2 方　　針

上記で述べた目標を解決するために，我々は以下のアプローチをとった．目標 (1) については，MLUTK をオブジェクト指向言語で実装することにより拡張性を担保し，さらに XML による設定記述を採用して，使用する言語理解モデルを自由に設定できるようにし

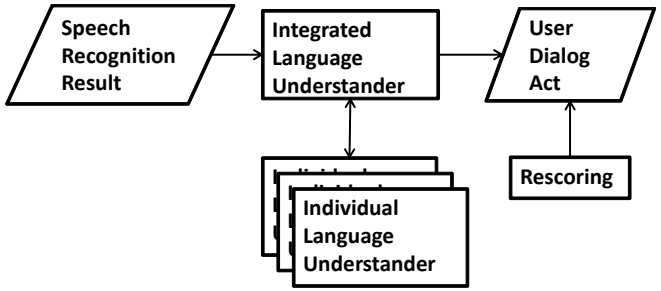


**図 3** MLUTK の基本構成

```
<config>
  <recognition>
    <grammar id="main" type="srgs" file="app/domains/main/main.grxml"/>
    <grammar id="dictation" type="n-gram" file="app/domains/main/ngram.grxml"/>
    <understander id="embed" class="hri.mlutk.lib.lu.SemanticTagExtractor"/>
    <understander id="crf" class="hri.mlutk.lib.lu.CRFExtractor"
                           accept-grammar="main"/>
  </recognition>
</config>
```

**図 4** 設定ファイルの記述例

た．XML は Extensible Markup Language のことで，人に読めるテキストで書かれていながら，コンピュータにも処理しやすいデータ・ベースの記述フォーマットであり， 広く使われている．また，目標 (2) で述べた課題の解決のために，共通の音声認識文法ファイル・フォーマットを設定し，学習用の訓練ファイル記述を XML に基づいて設計した．以下，これらについて詳述する．

### 3.3 オブジェクト指向言語による実装

オブジェクト指向プログラミング言語は，現在多くのソフトウェアの開発現場で使われていて，多くの解説書が世にある．オブジェクト指向言語の一般的な特徴の一つに多態性があるが，ここではこの多態性が特に重要である．端的にいって多態性とは，データの操作をデータの中身が何であれ，同じインターフェースで扱えるようにするための仕組みのことである．またオブジェクト指向では，ここでいう任意のデータや変数またはそれらの集合を一般化してオブジェクトと言い，オブジェクトのプログラムによる記述をクラスと呼ぶ．多態性によってオブジェクトの中身，すなわち言語理解モデルが何であっても，同じインターフェースを通して操作できるようになる．

MLUTK の基本構成を**図 3** に示す．音声認識結果は，“SpeechRecognitionResult” というデータとして表され，これには複数の言語モデルによる複数の認識結果が含まれている．そして “IntegratedLanguageUnderstander” によって意味理解結果を表す “UserDialogAct” に変換される．この時 “IntegratedLanguageUnderstander” は，複数の “IndividualLanguageUnderstander” を呼び出して変換結果を作るが，この複数の “IndividualLanguageUnderstander” は共通のインターフェース **understand** を持つ．このインターフェース **understand** の入出力仕様は固定されるが，その実装はそれぞれの言語理解モデルごとにカスタマイズすることができる．“UserDialogAct” には複数の言語モデルと言語理解モデルによる言語理解結果のリストが格納されている．それぞれの言語理解結果には音声認識のコンフィデンス値や言語理解モデルが出力する確率値などの値も格納されていて，これらを使ってそれぞれの理解結果に総合的なスコアを与える “Rescoring” という処理を行う．この “Rescoring” の処理も多態性を使って開発者がカスタマイズできるようになっている．このように，オブジェクト指向言語の多態性によって開発者が独自の言語モデルを実装したり，それらを自由に組み合わせたりすることが可能になる．実装上は，図 3 に示したフローチャート上のデータも処理も，それぞれオブジェクトのクラスとして記述されている．また，いくつかのオブジェクト指向言語はリフレクションと呼ばれる機能を提供しており，これによりプログラム全体を再コンパイルしなくても，後からオブジェクトを追加できる．

MLUTK を実装するに当たって，我々はオブジェクト指向言語として広く使われいる Java[⋆1] を選択した．Java で書かれたプログラムは多くのプラットフォームで動作し，リフレクション機能を持つオブジェクト指向言語でもある． さらに開発環境が整備されており，Weka[8] や MALLET[9] といった言語理解モデルの実装に必要な機械学習のプログラムも Java で多く開発されている．Java の利点の一つは，これらの既存のプログラムを自分のアプリケーションに組み込むことが非常に容易なことである．また，XML を扱うのに便利なクラスも Java のソフトウェア開発キットに含まれている．C# も Java と似たような機能を持っているが，Java ほどには広く普及していないため目的にあった機械学習の既存ライブラリを見つけるのは難しい．C++は仮想マシンを持たないオブジェクト指向言語であるの

⋆1 Java は、Oracle Corporation 及びその子会社、関連会社の商標です.

```
<grammar>
  <rule id="time">
    <one-of>
      <item><token sym="13" slot="startTime"/>午後 1 時</token></item>
      <item><token sym="14" slot="startTime"/>午後 2 時</token></item>
    </one-of>
  </rule>
</grammar>
```

**図 5** Galatea Toolkit での SRGS の意味タグ表記例

で，浮動小数点数の処理を大量に行う際にはもっとも実行速度が速い．しかしリフレクション機能を持たないため Java のような使い方はできない．Java は，Ruby，Python などのスクリプト系オブジェクト指向言語よりは実行速度が速く，今現在もっともバランスの取れた使いやすいオブジェクト指向言語である．

作成した MLUTK では**図 4** に示すような XML 形式の設定ファイルによって，使用する音声認識文法や言語理解モデルを実装したクラスを設定する．図 4 にあるように，recognition エレメントの下の grammar エレメントに使用する音声認識文法ファイルや大語彙温泉認識言語モデルを指定する．この grammar エレメントの “id” アトリビュートに指定された値（文字列）が音声認識の言語モデルを表す ID になる．また understander エレメントに使用する言語理解モデルを実装したクラス名を設定する．これは具体的には図 3 に示した “IndividualLanguageUnderstander” の実装クラスの名前である．grammar の場合と同じように，ここで指定した id が使用する言語理解モデルの ID となる．また，言語理解モデルごとに必要な設定などもこのエレメントに記述して利用できる．従って，例えば CRF の特徴量などに関する設定等も記述できる．MLUTK には，この設定ファイルを読み込んで，自動で設定されているクラスをインスタンス（メモリー上でのオブジェクトの実体）化し，さらに文法ファイルの読み込みをサポートする仕組みも実装されている．

MLUTK にはサンプルとして以下の言語理解モデルが実装されている．

- SemanticTagExtractor: 音声認識器に組み込みの言語理解結果を抽出する．
- CRFConceptExtractor： CRF を使った言語理解
- KeyphraseExtractor: キーワード抽出による言語理解

### 3.4 文　　法

MLUTK では，前に述べた目標 (2) に従って，音声認識と言語理解とを統一的に管理でき

```
<grammar>
  <rule id="city">
   <tag>out.nagoya = 0; out.wako = 5;</tag>
    <item repeat="0-1">名古屋<tag>out.nagoya = 1;</tag></item>
    <item repeat="0-1>和光<tag>out.wako = 2;</tag></item>
    <tag>out.city = out.nagoya + out.wako;</tag>
  </rule>
</grammar>
```

**図 6** SRGS に SISR によって意味タグをつけた例

```
<grammar>
  <rule id="main">
       <item>
      <ruleref uri="#time><tag>out.startTime=rules.time;</tag>
      </item>
      から
      <item>
          <ruleref uri="#time><tag>out.endTime=rules.time;</tag>
      </item>
      まで
  </rule>

  <rule id="time">
    <one-of>
      <item>午後 1 時<tag>out=13;</tag></item>
      <item>午後 2 時<tag>out=14;</tag></item>
    </one-of>
  </rule>
</grammar>
```

**図 7** MLUTK でサポートされている SISR による意味タグ付

るようにしたい．そこで意味タグ付きの音声認識文法をサポートすることにした．音声認識文法記述には様々なものがあるが，MLUTK では SRGS（Speech Recoginition Grammar Specification）[10] をサポートする．SRGS は W3C[⋆1]によって標準化され，XML に準拠しており音声認識用文法として必要十分な機能を含んでいる．Java としては，別に JSGF（JSpeech Grammar Format）[11] という非 XML 形式の音声認識文法がサポートされているが，MLUTK では XML を標準とすることにした．また，SRGS は VoiceXML[12)⋆2]の標準の音声認識文法フォーマットであり，SRGS の XML 形式をサポートする市販の音声認識器もある[7]．ただ MLUTK では JSGF も使えるように変換ツールを整備している．

言語理解を行うためには，音声認識文法に意味タグをつける必要がある．意味タグとは，例えば「午後 1 時」というユーザ発話をビデオ予約の開始時間が「13 時」であると解釈するためのものである．大抵これは，フレームのスロットとして処理される．この場合はスロット名が「録画開始時間」（startTime）で，そしてスロットの属性に「13」がはいることになる．Galatea Toolkit[6] では，SRGS を拡張して**図 5** のように，token エレメントの slot アトリビュートに スロット名を記述するようになっている．この場合，token エレメントのテキストノードが，このスロットに入り得る属性の一つになる．しかしながら，これでは同じ属性を開始時間だけでなく，ビデオ録画の終了時間にも使いたい場合，同じことをスロット名だけ変えて書かなければならなくなる．一方これには利点もあって，スロット名とその属性の関係が単純であるので，比較的単純な言語理解モデルを実装しやすい．

W3C では SRGS とあわせて SISR(Semantic Interpretation for Speech Recognition)[13] という意味タグ書式が勧告されている．これは，ECMAScript[14)⋆3]によって意味タグを記述するものであり非常に高機能である．スロット名は変数名となり，属性はその変数の内容として表される（**図 6**）．この意味タグ付の仕組みは多少複雑で，音声認識結果に応じてスクリプトが解釈され，意味タグが生成される[⋆4]．これを使えばスロット名ごとに同じルールを書かなくても済むという利点もあり，特に数字の解釈には効果的である．さらに，SISR では関数を定義できたり，もっと高度な処理も可能である．しかしながら，高機能であるが故に単語ごとのタグ付を基本とする単純な言語理解手法とミスマッチが生じる場合がある．図 6 の例でいえば,「名古屋」と「和光」がそろって発話されたからスロット名 city の属性が 3 になる等の演算処理を MLMU で使用する全ての言語理解モデルに組み込まないといけなくなる．MLMU で使用する言語理解モデル全てに，このような高度な処理は必要ではなく，SISR は高機能すぎるのである．また MLMU は大語彙音声認識結果も使うため，常に音声認識文法に書かれた SISR が解釈可能になるわけではないことにも留意する必要もある．つまり言語モデルや音声理解モデルごとに，意味タグの記述ファイルを用意するようにするのは，前に述べた MLUTK の目標 (2) に反する．一つの意味タグ付きの音声認識文法記述で,（それを自動で変換する等して）音声認識を行い，また意図通りの言語理解ができることが望ましい．

以上のことから，MLUTK では SISR による意味タグの記述を使うが，機能を大幅に制限した SISR を意味タグの記述に用いる。具体的には，変数への代入演算のみをサポートする。これによって，スロット名と属性の関係の単純化と，スロット名の別名での利用を可能にできる．例を**図 7** に示す．この文法記述では，ユーザの「午後 1 時から午後 2 時まで」という発話があるときスロット名 startTime に属性「13」がセットされ，スロット名 endTime に「14」がセットされる．また,「午後 1 時」あるいは「午後 2 時」という発話内容のスロット名は startTime か endTime かのどちらかである．

### 3.5 訓練データ

言語理解モデルの訓練データには，以上の音声認識文法記述と整合性が取れるような記述形式が求められる．例えば CRF で系列ラベリングを行う場合，まず単語ごとにタグ付された例文で学習する必要がある．MLUTK では二つのタグ付方法を提供する．一つは全自動でタグ付された例文を，意味タグ付きの SRGS から生成する．この場合は SRGS に記述された全ての発話パターンをタグ付けされた例文として生成する．ここでのタグは，スロット名である．例えば SRGS の記述が図 7 の場合，**図 8** のように行ごとにタグ付された例文が出力される．このとき**図 9** のような単語クラスの設定ファイルを用いることで，単語ごとに追加の「時刻」などの属性も自動でつけることができる．もう一つの選択肢は，**図 10** の

---

⋆1 W3C は、World Wide Web Consortium の商標です.
⋆2 VoiceXML は VoiceXML Forum の商標です.
⋆3 ECMAScript は Ecma International の商標です.
⋆4 SRGS+SISR で，どのように意味タグが生成されるか補足する．図 6 のように音声認識文法が書かれていて，ユーザ発話が「名古屋」であった場合，まず,

```
<tag>out.nagoya = 0; out.wako = 5;</tag>
```

の部分が解釈されて，変数 out.nagoya は 0 にセットされ，out.wako は 5 にセットされる．そして，ユーザ発話は「名古屋」だけで,「和光」を含まないので

```
<tag>out.nagoya = 1;</tag>
```

だけが実行される．そして結果として

```
<tag>out.city = out.nagoya + out.wako;</tag>
```

によって，変数 out.city の値が 1+5 で 6 になり，この値がスロット名 city の属性になる．

```
午後 1 時　時刻　startTime
から　　null
午後 2 時　時刻 endTime
まで　 null

午後 1 時　時刻　startTime
から　　null
午後 1 時　　時刻　 endTime
まで　 null
```

**図 8**　タグ付された例文

```
<semantics>
  <class name="時刻">
    <exp>(午前|午後)?([0-9]|1[0-9]|2[0-4]) 時</exp>
  </class>
</semantics>
```

**図 9**　単語クラスの設定

```
<train config="config.xml">
  <action id="videoReq">
    <exp>開始時間は 午後 1 時 です</exp>
    <exp>終了時刻は 午後 2 時 です</exp>
    <exp>{午後 1 時:startTime} から {午後 2 時:endTime} まで</exp>
  </action>
</train>
```

**図 10**　訓練データ記述の例

ように例文は人が記述するものである．こちらの方はキーワードを検出して，自動でタグ付も行うが，キーワード検出で間違える可能性のあるものについては，人手でタグ付けする．例えば「午後 1 時」とか「午後 2 時」のような，複数のスロット名がある場合である．もちろん，これ以外に音声認識文法とは別個に人手でタグ付けを行う言語理解モデルがあってもよい．しかしながら，その場合は学習用のタグ付と意味タグつきの音声認識文法とを別個に管理しなければならない．MLUTK では，これらの言語理解の訓練用の記述を使って，クラス n-gram を作成するツールもある．従って音声認識文法と学習用訓練ファイルとを記述すれば，n-gram による音声認識のための言語モデルも作成でき，かつ言語理解モデルの学習も行えるようになっている．

## 4. 考　　察

MLUTK は標準的な Java 環境で動作するため，PC だけでなく Android[⋆1] 上でも動作する． 従って携帯端末等でも高度な音声理解ができる可能性がある．また，サンプルとして CRF による言語理解モデルを実装したが，Java で実装されている MALLET を使うことで，比較的簡単に実装できた．もちろん RIME-TK と組み合わせて，音声対話システムを構築することもできる．

また音声認識文法の意味タグ付に SISR を採用したので，スロット名のための冗長な記述を避けることができる．さらに SISR の代入演算のみをサポートすることで，音声認識文法を記述さえすれば，キーフレーズ抽出による言語理解モデルが動作する．この音声認識文法記述から，簡単な学習のためのデータを作成することも可能であり，n-gram による言語モデルも作成できる．これに例文を学習用訓練ファイルに記述すれば，さらに頑健な言語理解モデルの学習や大語彙音声認識ができるようになる．

以上のように MLUTK は高度な音声認識・理解を備えた音声対話システム構築のためのツールと方法論一式を提供するものである．

## 5. おわりに

本報告では，我々が作成した複数の言語理解モデルをオンラインで動作する音声対話システムに容易に組み込むためのツールキット（MLUTK）の概要について述べた．

今後の予定としては，言語理解結果の選択の部分についても対象を拡張し，CMBS など

⋆1 Android は Google Inc. の商標です．

による選択手法も組み入れていきたい．またマルチドメイン対話システムにおける有効性も検証していきたい．

## 参 考 文 献

1) 勝丸真樹，中野幹生，駒谷和範，船越孝太郎，辻野広司，尾形哲也，奥乃　博：複数の言語モデルと言語理解モデルによる音声理解の高精度化，電子情報通信学会論文誌，Vol.J93-D, No.6, pp.879–888 (2010).
2) 勝丸真樹，駒谷和範，中野幹生，船越孝太郎，辻野広司，尾形哲也，奥乃　博：複数の言語モデルと言語理解モデルによるラピッドプロトタイピング向け音声理解，情報処理学会研究報告, Vol.2009-SLP-80，No.5, pp.1–6，(2010).
3) Raymond, C. and Riccardi, G.: Generative and Discriminative Algorithms for Spoken Language Understanding, *Proceedings of INTERSPEECH 2007*, pp.1605–1608 (2007).
4) Nakano, M., Hasegawa, Y., Funakoshi, K., Takeuchi, J., Torii, T., Nakadai, K., Kanda, N., Komatani, K., Okuno, H.G. and Tsujino, H.: A multi-expert model for dialogue and behavior control of conversational robots and agents, *Know.-Based Syst.*, Vol.24, pp.248–256, (2011).
5) Alam, H., Rahman, A. F. R., Tjahjadi, T., Cheng, H., Llido, P., Kumar, A., Hartono, R., Tarnikova, Y. and Wilcox, C.: Development of Spoken Language User Interfaces: A Tool Kit Approach., *SSPR/SPR'02*, pp.339–347 (2002).
6) 嵯峨山茂樹，川本真一，　下平博，新田恒雄，西本卓也，　中村哲，伊藤克亘，森島繁生，四倉達夫，甲斐充彦，　李晃伸，山下洋一，小林隆夫，徳田恵一，広瀬啓吉，峯松信明，　山田篤，　伝康晴，宇津呂武仁：擬人化音声対話エージェントツールキット Galatea, 情報処理学会研究報告,2002-SLP-45-10，pp.57–64 (2003).
7) Nuance Recognizer9 [Online] `http://www.nuance-jp.com/product/NuanceRecognizer.html`.
8) Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and H., I.: The WEKA Data Mining Software: An Update, Technical Report Volume 11, Issue 1, SIGKDD Explorations (2009).
9) McCallum, A.K.: MALLET: A Machine Learning for Language Toolkit. [Online] `http://mallet.cs.umass.edu`.
10) Speech Recognition Grammar Specification Version 1.0. [Online] `http://www.w3.org/TR/speech-grammar/`.
11) JSpeech Grammar Format [Online] `http://www.w3.org/TR/jsgf/`.
12) Voice Extensible Markup Language (VoiceXML) 2.1 [Online] `http://www.w3.org/TR/voicexml21/`.
13) Semantic Interpretation for Speech Recognition (SISR) Version 1.0. [Online] `http://www.w3.org/TR/semantic-interpretation/`.
14) ECMAScript. [Online] `http://www.ecma-international.org/publications/standards/Ecma-262.htm`.