チャットで話す「冷蔵庫番」

竹　形　誠　司†

近年，スマートフォンを代表とする携帯型通信機器は目覚しい発展を遂げている．中でも音声認識技術の進化は，自然言語インターフェイスの実用化に大きく寄与し，モバイルアプリケーションの設計手法に大きな影響を与えるものと期待される．本稿では，筆者らが開発中の自然言語インターフェイスを備えたアプリケーションフレームワーク「YouBot」を用いて，冷蔵庫の内容に関する問い合わせに答えるアプリケーションを作成する方法について述べる．

Talking to a Fridge Keeper

Seiji Takegata†

During the past several years, there has been a dramatic evolution in mobile communication devices such as smart phones. Especially, speech recognition technology is contributing a lot to practical implementation of natural language interface and it will have a big impact on mobile application design. In this paper, we describe how to create an application which remembers the foods in the refrigerator and answers to a query from a user using our NLI application framework YouBot.

1. はじめに

近年の携帯型通信端末は高性能化，高機能化によってマルチタッチやマルチタスクなどの機構を利用した高度なユーザインターフェイスを備えるようになった．しかし，文字の入力に関してはいまだに課題が多い．一般にスマートフォンと呼ばれるタイプの携帯端末では，3.5〜4 インチのタッチスクリーン式のディスプレイを備えるものが多いが，QWERTY 式のフルキーボードを操作するにはこの画面では小さく，使い易いものとは言えない．フリックやスワイプなど新たな入力方式も登場しているが，従来の日本の携帯電話のように誰もが使える操作性を実現するまでには至っていない．そこで注目されるのが音声による入力である．

これまでにも音声認識による文字入力は試されてきたが，初期の製品で認識精度が低かったり，話者ごとにソフトウェアをトレーニングする必要があったりしたため，一部を除いて実用には程遠いというイメージが根強く残っていた．しかし，Google が公開した音声検索ソフトウェアが高い認識精度を示したことで，状況は変わりつつある．最新の Androd OS ではソフトウェアキーボード機能の一部として音声認識が組み込まれており，この OS 上ではどのソフトウェアも音声入力が使えるようになった．

図 1　冷蔵庫番

以前より筆者らは自然言語インターフェイスの可能性について研究を続けており，チャット型のインターフェイスを備えたアプリケーションフレームワークである「YouBot☆」を開発中である[1),2)]．このフレームワークでは，JavaScript を始めとする簡易的なスクリプト言語でアプリケーションを作成することができ，スプリクトを追加することにより柔軟に機能を拡張することができる．スクリプトは，ユーザの入力した文

† 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, University of Tokyo

☆ http://youbot.world-lab.net

を単にデータの入力として扱うのではなく，データを含んだシステムへの指令として解釈し，質問に応答したり，データを更新したりする．電話を兼ねた端末では，ユーザが端末に対して音声で語りかけることは極めて自然であり，自然言語インターフェイスとの親和性は非常に高いと考えられる．

本稿では，冷蔵庫内の食品に関して，ユーザからの問い合わせに答えるアプリケーションを自然言語インターフェイスによって実現する方法について述べる．また，音声対話に対応した端末側プログラムの試作についても若干触れる．

2. YouBot フレームワーク

2.1 ユーザとボット及びボット間の通信

YouBot はネットワーク上で IM(インスタントメッセージング) クライアントとして動作する．ユーザは端末上で別の IM クライアントを起動し，YouBot に接続して対話を行う．YouBot は一般的な対話エージェント[3] とは異なり，1 つのボットは特定のユーザだけを対象とする．ボット同士はボット間通信と呼ばれるプロトコルを用いて相互に情報を提供する．ユーザ同士はこの仕組を使って間接的に情報を交換することができる．本稿では，ボット間通信を使って冷蔵庫の情報を入手する構成とした (図 2)．

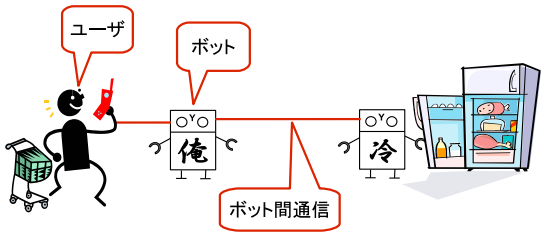

図 2 ユーザとボット及びボット間の通信

2.2 フレームワークとアプリケーション

YouBot におけるフレームワークとアプリケーションの関係は図 3 のようになっている．フレームワークはユーザのメッセージを解釈するためのパターン照合器や、情報収集のためのウェブページアクセス機能、収集した情報を格納するためのデータベースなどを備えている。また、twitter から情報を収集することも可能である。アプリケーションはタスクプロセッサと呼ばれる追加可能なスクリプトモジュールによって構成される．フレームワーク自体は Java 環境に構築されており，スクリプトの実行環境は，標準の ScriptingEngine を利用している．現在は JavaScript でタスクプロセッサを記述しているが，原理的には ScriptingEngine が対応するすべてのスクリプティング言語が使用可能である．また，ScriptingEngine に対応した新たな言語モジュールを開発することにより，対話処理に特化した独自のスクリプティング言語を設計することもできる．ユーザやユーザのツイート，他のボットからのリクエストはタスクマネージャが受け取り，タスクプロセッサに転送する．

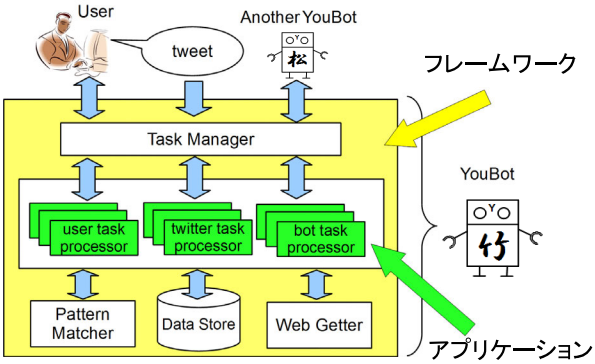

図 3 フレームワークとアプリケーション

2.3 タスクマネージャとタスクプロセッサ

図 4 はタスクマネージャとタスクプロセッサ間の処理の受け渡しを表したものである．タスクマネージャはユーザや他のボットから受け取ったメッセージを引数として，すべてのタスクプロセッサの estimate 関数を呼び出す．タスクプロセッサの estimate 関数では，渡されたメッセージをどの程度よく解釈できるかを判断し，スコアを返す．フレームワークはこれらのスコアを回収し，最も高いスコアを提示したタスクプロセッサの process 関数を呼び出す．呼び出されたタスクプロセッサの process 関数はメッセージに応じた処理を行い，ユーザへの応答文を作成する．つまり，本フレームワークでは，ユーザのメッセージをタスクマネージャで解釈する方式[4] ではなく，各モジュール側で解釈する方式[5] を取っている．

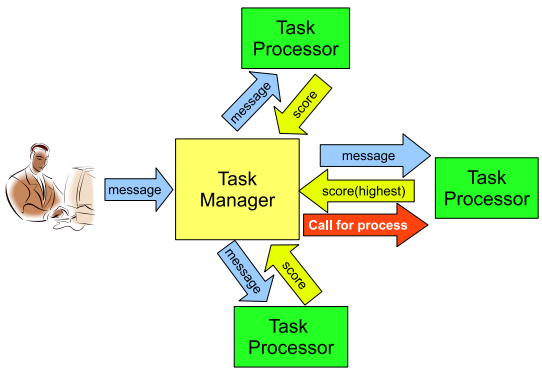

図 4 タスクマネージャとタスクプロセッサ

3. タスクプロセッサの記述例

ここでは冷蔵庫番アプリケーションを例にタスクプロセッサの記述方法について説明する．冷蔵庫番アプリケーションは，問い合わせるユーザ側のボットで AskFridge スクリプト，応答する側のボットで AnswerFridge スクリプトが実行される．

3.1 AskFridge

次のコードは AskFridge スクリプトの estimate 関数である．PatternMatcher オブジェクト (pm) に文字列とパターンをセットして match() 関数を呼ぶと，マッチした場合はハッシュマップオブジェクトを，マッチしなかった場合は null を返す．パターンの記述に使われる“ ( ) ”は省略可能な選択肢，“ { } ”は必須の選択肢，“ [ ] ”はラベル付きの任意の文字列である．このラベル付きの任意の文字列は，パターンがマッチした場合に HashMap オブジェクト (hm) に格納され，ラベルをキーに文字列を取得することができる．match 関数から返された値が null でなければ 10 を返し，null の場合は 0 を返す．

```
function estimage(message){
  pm=PatternMatcher.getNew();
  pm.setString(message);
  strPattern="冷蔵庫に(は)[what](が|は)"
    +"{ある(かな)|あった(かな)}(？)";
  pm.setPattern(strPattern);
  hm=pm.match();
  if(hm != null){
    score=10;
  }else{
    score=0;
  }
  return score;
}
```

この PatternMatcher には，サブパターンを定義する機能があり，日付や時刻などのサブパターンを定義して上位のパターンに組み込むことができる．

次のコードは AskFridge スクリプトの process 関数である．HashMap オブジェクトから“［what］”をキーに文字列を取得し，問い合わせメッセージに組み込んでいる．問い合わせメッセージは inquiry オブジェクトにセットされ，相手のボットに送信される．

```
function process(){
  strWhat= hm.get("[what]");
  message="ASK_FRIDGE:"+strWhat;
  inquiry=task.getNewInquiry();
  inquiry.setMessage(message);
  inquiry.setDestination(botAddress);
  inquiry.send();
}
```

フレームワークが相手のボットから返信を受け取ると，タスクプロセッサの convey 関数を呼び出す．次の convey 関数では，渡された inquiry オブジェクトから応答メッセージを取得し，task オブジェクトの say 関数でユーザのクライアントに向けて送信している．

```
function convey(inquiry){
  message = inquiry.getResponse();
  task.say(message);
}
```

所定の時間を経過しても相手のボットからの応答がない場合は timeout 関数が呼び出される．ここでは次のように timeout 関数を定義する．

```
function timeout(){
  task.say("応答がありません");
}
```

3.2 AnswerFridge

フレームワークがボットからのリクエストを受信した場合は estimate 関数に inquiry オブジェクトが渡される．問い合わせを受け付ける側の estimate 関数は次のようになる．

```
function estimate(inquiry){
  message = inquiry.getMessage();
  block = message.split(":");
  if(block[0]=="ASK_FRIDGE"){
    what = block[1];
    score=10
  }else{
    score=0;
  }
  return score;
}
```

YouBot のデータオブジェクトは key=value 形式のデータストアである．次の例は「冷蔵庫に卵が 3 個保存されている」ことを表すデータである．

```
type=fridge
item=卵
qty=3
unit=個
```

応答文を作成する process 関数は次のようになる．まず，selector オブジェクトにフィルタ条件を指定してデータストアからデータオブジェクトのリストを抽出する．下の例ではリストのサイズが 0 でなかった場合に先頭の要素から qty(数量) と unit(単位) を取り出して応答文に組み込んでいる (紙幅に制限があるためアイテムの重複を考慮していない)．

```
function process(){
  selector = task.getNewDataSelector();
```

```
  selector.addFilter("type","fridge");
  selector.addFilter("item",what);
  listItems = selector.getList();
  if(listItems!=0){
    item=listItems.get(0);
    response=item.get("qty")
      +item.get("unit")+"あります";
  }else{
    response="ありません";
  }
  inquiry.setResponse(response);
  inquiry.respond();
}
```

3.3 ユーザとボットの対話

これらのスクリプトにより実現されるユーザとボットの対話の例を次に示す.

```
USER>冷蔵庫に卵はあったかな
BOT >3つあります
USER>冷蔵庫に牛乳あるかな？
BOT >ありません
```

現在テストしているスクリプトは，上記のスクリプトを改良して，2 度目の問い合わせからは「冷蔵庫に」を省略しても正しく解釈されるようにしている．また，食品を「野菜」や「飲み物」などのグループに分類して，グループ単位で問い合わせができるようにしている.

4. モバイル端末の利用

本研究では文字列によるコミュニケーションを主眼としているため，音声入力については対象外としてきたがモバイル環境における音声認識入力の有用性を確認するために，音声入出力に対応した IM クライアントを試験的に実装した．実行環境は Google 社の Android OS v2.2 である．前述の通り，音声認識に関してはソフトウェアキーボードの一部として既に組み込まれている．Android OS には音声合成のための API も存在するが，日本語に関しては漢字表記から「読み」に変換する必要があり，現在の所では標準では対応されていない．そこで今回はクリエートシステム開発株式会社☆のアプリケーション「ドキュメントトーカ」を使用した．ドキュメントトーカは API が公開されており，外部のアプリケーションから読み上げ機能を利用することができる．IM のプロトコルには XMPP☆☆を採用し，ライブラリに ignite realtime 社☆☆☆の smack を利用した.

音声認識，音声合成の精度，通信のレスポンスは概ね良好である．屋外での使用にはヘッドセットを利用することで環境雑音の影響を低減できる．Bluetooth を利用する場合は通話用のヘッドセットではオーディオ再生に対応していないものが多く，音楽再生に対応したヘッドセット (A2DP など) を用意する必要がある.

音声認識の傾向として，書き言葉に近い文は比較的正確に変換されるが，会話のようなくだけた文では誤変換の確率が高まるようである．これは変換の計算に使われているコーパスの偏りによるものであると考えられる．今後，Google 社のシステムに話し言葉のデータが蓄積され，これらの問題が解決されることを期待する.

5. おわりに

本システムが正しく応答できるのは予め用意されたパターンにマッチする文が入力された場合だけであり，いかに多くのスクリプトが作成されるかがプロジェクトの成否を左右する最大のポイントである．本稿執筆時点ではプログラムを公開できる段階に至っていないが，本プロジェクトはオープンソース化を目指しており，プログラム公開の暁には幅広くユーザの意見を求めていく所存である.

参 考 文 献

1) 竹形誠司, 田中久美子：チャット形式で対話するネットワーキング秘書ボット「youbot」，言語処理学会大会第 16 回年次大会論文集, pp.210–213 (2010).
2) S. Takegata, K. Tanaka-Ishii：YouBot: A Simple Framework for Building Virtual Networking Agents, *In Proceedings of the 11th annual SIGdial Meeting on Discourse and Dialogue*, pp. 273–276 (2010).
3) R.S. Wallace. The Anatomy of A.L.I.C.E. *Parsing the Turing Test*, pages 181–210, Springer Netherlands, 2008.
4) A. Nguyen. An agent-based approach to dialogue management in personal assistants. *In Proceedings of IUI-2005*, pages 137–144. ACM Press, 2005.
5) M. Nakano, K. Funakoshi, Y. Hasegawa, and H. Tsujino. A Framework for Building Conversational Agents Based on a Multi-Expert Model. *In Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 88–91. ACL, 2008.

☆ http://www.createsystem.co.jp
☆☆ http://xmpp.org/
☆☆☆ http://www.igniterealtime.org/