

語彙階層とオブジェクト階層を利用したスマートスピーカの短文解釈方式の提案

本多 佑希¹ 岸本 有生² 兼宗 進¹

概要：スマートスピーカの音声入力に対する応答をプログラムできる学習環境の開発を進めている。本発表では、「すずめは飛ぶ？」や「犬の鳴き声」といった短文から「すずめ」「飛ぶ」「犬」や「鳴き声」などの語彙を抽出し、すずめが鳥の一部であることや、犬が動物の一部であるといった語彙の階層構造をプログラムのオブジェクト階層に対応付けて扱う拡張について報告する。自然言語である入力文に対する解釈の妥当性や人工言語であるプログラミング言語でのルール表現の妥当性を確認するために、大学生を対象とした小規模な実験授業での評価を行なった。

1. はじめに

スマートスピーカのアプリケーション開発を支援するプログラミング学習環境の開発を進めている。スマートスピーカーはユーザが音声によってアプリケーションの起動や操作を行うデバイスである。「10時に起こして」といった文章から「10時」や「起こして」といった語句をAIを用いた音声認識により抽出し、適切な処理を行ってユーザに応答する。

著者らは過去に、「大阪」「洋食」「暑い」などの関数名で処理を定義することで、スマートスピーカーに話しかけた単語に応じて適切な処理を行う環境 [1] を提案してきた。単語のみの認識に限定することでプログラムの記述法の難易度を下げたことで、高校生でも十分に学習が可能になったことを確認している。

そこで、次の段階として単語のみの認識ではなく、「今日の天気は？」といった短文を認識し、「今日」や「天気」といった単語を切り出してプログラム上から処理する手法を検討した。また、「すずめは飛ぶ？」や「ペンギンは飛ぶ？」といった文章から、「すずめ」と「ペンギン」はそれぞれ「鳥」というグループに含まれるといった語彙の階層構造に着目した。語彙の階層構造とプログラミング言語のオブジェクトの継承関係の階層構造が類似しているため、オブジェクト継承を用いることで短文の記述を簡略化できるのではないかと考えた。

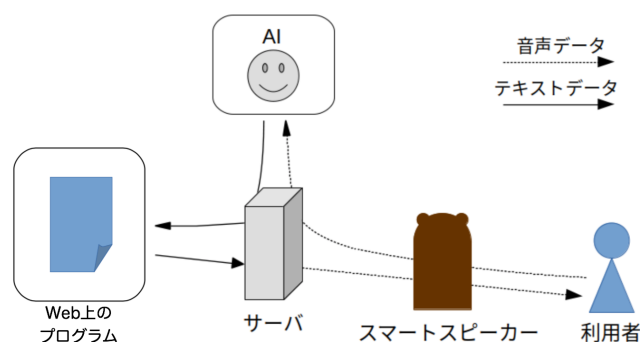


図1 一般的なスマートスピーカのシステムモデル

検討した記述方式を用いて、大学生を対象に実験的に授業を行った。ルールの妥当性を確認するとともに、記述方式を応用して、学生が自分でイメージしたモデルをプログラムに落とし込むことができるかを確認した。本稿では記述方式を提案するとともに、授業の結果について報告する。

2. スマートスピーカー

スマートスピーカーはユーザが音声によりアプリケーションの起動や操作を行う機器である。ユーザが発した音声をサーバに送信し、サーバが音声認識エンジン (AI) を用いてテキストに変換し、適切な処理を行う。システムを簡略化した概要図を図1に示す。

3. 提案する記述方式

3.1 問題

図2に、著者らが開発を進めていた、スマートスピーカーが認識した単語によって動作するプログラムの例を示す [1]。記述するプログラミング言語は、ドリトルを採用し

¹ 大阪電気通信大学
Osaka Electro-Communication University

² 大阪電気通信大学高等学校
Osaka Electro-Communication University High School

```

1 暑い=「
2  クローバー！"アイスを食べましょう。"話す。
3  ♪
4  寒い=「
5  クローバー！"コーンスープを飲みましょう。"話す。
6  ♪

```

図 2 単語の認識に用いたプログラムの記述方式

ている [2]。「暑い」という単語を認識したら、「暑い」という命令が実行される形式である。イベントの登録を、命令の定義によって表現しているイメージである。単語をそのまま命令名として記述することによって、標準的な SDK を使うプログラムに比べ、大幅にプログラムの難易度を下げることができた。しかし、「天気」や「暑い」などといった単語は認識できても、「今日の天気」のような文章を認識することはできない。こういった文章が認識できれば、より実用的なプログラムの学習が可能になる。

3.2 認識対象とする文の検討

今回認識の対象とする文章について検討を行った。スマートスピーカーのアプリケーションの起動や操作のために話しかける文章の中で、代表的なものを以下に示す。

- 今日の天気は？
- 雨は降る？
- 株価を教えて
- 10時に起こして
- 音楽をかけて

スマートスピーカーには多種多様なアプリケーションが存在する。それぞれのアプリごとに起動・操作のためのテキストは異なる。しかし、上記のリストから、以下のような形の短文を認識できれば基本的なスマートスピーカーの操作を表現することができるのではないかと考えた。

- (1) A の B は？
- (2) A は B？
- (3) A を B

3.3 プログラムでの表現方法の検討

今回認識する対象とする文章は 3.2 章で述べた通りである。これらの文章をプログラム上で表現するために、図 3 の記述方式を検討した。

「すずめは飛ぶ？」という文章を認識した際の動作は、「すずめ」オブジェクトに「飛ぶ？」という命令として定義する。これにより、「A は B？」という文章を表現する。

「A の B は？」という文章も、例えば「今日の天気は？」をプログラム上で表現する際には、「今日」オブジェクトに「天気？」という命令として定義する。「天気」はあくまで名詞であるが、「？」をつけることによって、天気を教えてほしいという動詞の形を省略する形で表現している。

```

1  すずめ=単語！作る。
2  すずめ:飛ぶ？="クローバー！"飛びます。"話す。♪
3  今日=単語！作る。
4  今日:天気？="クローバー！"雨が降るでしょう。"話す。♪
5  夕食=単語！作る。
6  夕食:考えて="クローバー！"カレーはいかが？"話す。♪

```

図 3 短文を認識した際の処理を定義するプログラム

「A を B」という文章も、同様に表現できる。例えば、「夕食を考えて」という夕食提案アプリを考えた際には、「夕食」オブジェクトに「考えて」という命令を定義することで表現する。

これらで共通して、A は名詞であり、B は動詞（ないしは動詞を省略した名詞、以下は単に動詞とする）であることがわかる。そのため、3.2 章で検討した内容と、本項での検討内容から、スマートスピーカーの起動・操作のためのテキストの多くは名詞をオブジェクト名、動詞を命令名として記述することで表現できるのではないかと考えた。

3.4 学習活動を通して得ることができる知識

学習者は、この記述方式のルールに則ってスマートスピーカーのアプリケーション開発を行う。このルールを理解し、自身が考えたモデルをプログラムで表現する学習を通して、オブジェクトにはどのような命令・プロパティを定義すれば良いかを考えることになる。普段使用している日本語の形式から名詞、動詞を抜き出してプログラムの形に落とし込むことができるため、自分の知識の応用でオブジェクトの設計に関する考え方を学習できると考えている。

オブジェクトの設計やオブジェクト指向の考え方の学習は初学者には敷居が高く、高校の授業では取り入れられていない。情報処理学会のカリキュラム標準 J17 [4] では、ソフトウェアエンジニアリング領域の中では「ソフトウェア構築」という講義形式の授業の第 4 回目で扱われている。また、コンピュータ科学領域の中では「プログラミング言語」というエリアの「オブジェクト指向プログラミング」ユニットの Tier1 のトピックスとしてオブジェクト指向設計やクラスが挙げられている他、「情報管理」というエリアの「データモデル」ユニットの Tier2 の中でオブジェクト指向モデルが挙げられている。

オブジェクト指向の技術や、クラス設計などの学習を目的とした研究としても多く扱われている。松澤ら [5] は新入社員研修の中でオブジェクト指向技術者を育成するためのカリキュラムを作成している。高井ら [6] は大学 3 年生を対象とした講義の中で、提示されたクラス図を基に Java のプログラムを記述する課題を出すなどの学習を行っている。

これらのことから、オブジェクト指向の考え方やクラス設計はプログラミング学習においては重要な要素であるこ

```

1 すずめ = 単語！作る。
2 すずめ:飛ぶ? =「
3   クローバー！"すずめは飛びます。" 話す。
4  ♂
5 ハト = 単語！作る。
6 ハト:飛ぶ? =「
7   クローバー！"ハトは飛びます。" 話す。
8  ♂
9 トキ = 単語！作る。
10 トキ:飛ぶ? =「
11   クローバー！"トキは飛びます。" 話す。
12  ♂
13 ツバメ = 単語！作る。
14 ツバメ:飛ぶ? =「
15   クローバー！"ツバメは飛びます。" 話す。
16  ♂

```

図 4 図 3 の前半を様々な種類の鳥に適用したパターン

とがわかる。また、これらの技術は大学の講義や企業の新人研修で学ぶ高度な技術であることがわかる。そのため、今回提案する方式でプログラミング初学者がこれらの技術を実習形式で体験的に学習できるということは、プログラミング教育においても大きな意味を持つと考えている。

4. オブジェクト階層を用いた語彙階層の表現

4.1 問題

3 章で提案した記述方式によって、短文を認識する。プログラムの構造や、使用する機能についてはオブジェクトの生成と命令定義しか行っていないため、初学者でも十分に理解できると考えられる。しかし、図 4 のようにすずめやハト、トキなど複数の鳥を認識するプログラムを記述した際には、同じような命令定義が複数回登場し、再利用性が考えられていないことがわかる。

4.2 検討

例に挙げているすずめやハト、トキやツバメなどは「鳥」という分類に含まれる。今回は、こういった「語彙の階層構造」に着目した。語彙の階層構造としては、広く使われているものとしてシソーラスが存在する [3]。シソーラスで扱われる上位語や下位語、広義語や狭義語といった考え方を参考にしながら簡易化することで、単語同士の関係をプログラム上で表現できるのではないかと考えた。この階層構造を図で表現したものを図 5 に示す。すずめやハトなどの単語が、鳥から派生した形になっている。

すずめやハトなどの単語が鳥から派生している構造を考えた際に、プログラミング言語の「オブジェクトの階層構造」に類似していることに気付いた。「鳥」というオブジェクトを「すずめ」や「ハト」などのオブジェクトが継承している。この「語彙の階層構造とオブジェクトの階層構造の類似点」を利用することで、語彙の階層構造をプログラムとして表現できると考えた。

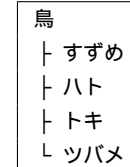


図 5 語彙の階層構造

```

1 鳥 = 単語！作る。
2 鳥:飛ぶ? =「|A|
3   クローバー！（A+は飛びます。）" 話す。
4  ♂
5 すずめ = 鳥！作る。
6 ハト = 鳥！作る。
7 トキ = 鳥！作る。
8 ツバメ = 鳥！作る。

```

図 6 図 4 を語彙階層とオブジェクト階層の考え方をを使って表現し直したプログラム

4.3 オブジェクトの階層構造

今回対象としているプログラミング言語「ドリトル」は JavaScript で知られる prototype ベースのオブジェクト指向を採用している。C++ などと違いクラスという概念はなく、オブジェクト同士がプログラム実行中に動的に継承関係を作る。

今回の場合、図 5 の「鳥」「すずめ」はそれぞれがオブジェクトであり、クラスとオブジェクトという関係ではない。そのため、今回のこの関係を図示する際には、オブジェクト図は不適切である。イメージではクラスベースの基底クラス、派生クラスという関係に近い。そのため、これらの関係を図示する場合にはクラス図の方が適切であると考えられる。したがって本稿ではこれらのオブジェクトの関係を示すためにクラス図を用いるが、厳密なものではなく、関係を示すためのものと認識されたい。

4.4 表現方法について

4.2 章で検討したように、今回は語彙の階層構造をオブジェクトの階層構造で表現することにした。図 6 に、図 4 を 4.2 章で検討した方式で表現し直したプログラムを示す。親オブジェクトである「鳥」のメソッドとして「飛ぶ」を定義しており、子オブジェクトである「すずめ」「ハト」などは継承関係を辿って「飛ぶ」を参照することができる。この方式により、再利用性が向上されたプログラムを実現することができた。

5. 授業実践

5.1 概要

3 章で検討した記述方式、及び 4 章で検討した階層構造のルール妥当性を検証するために、工学部の大学 3 年生 11 名を対象に実験的な授業を行った。対象の学生はドリトルの講習を事前に受けており、ドリトル言語の構文につ

```

1 鳥 = 単語！作る。
2 鳥:飛ぶ？ = 「|A|
3   クローバー！（A+「は飛びます。」）話す。
4  ♂
5  すずめ = 鳥！作る。
6  ハト = 鳥！作る。
7  トキ = 鳥！作る。
8  ペンギン = 鳥！作る。
9  ペンギン:飛ぶ？ = 「
10   クローバー！"ペンギンは飛ばない"話す。
11  ♀

```

図 7 授業内で使用した演習 1 の内容

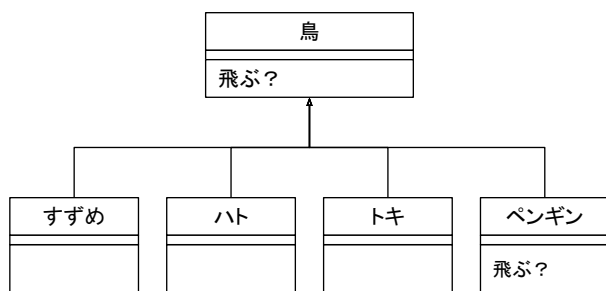


図 8 図 7 の階層構造

いては熟知している。また、C や Python などの知識もあるため、プログラミングについてはある程度習熟している。

5.2 授業内容

対象の学生には、基本的な記述方式のレクチャーを行ったあと、「簡単な演習と、演習の内容を踏まえた課題を解く」という活動を 2 セット行ってもらった。

1 回目の演習で用いたプログラムを図 7 に示す。「鳥」という基底のオブジェクトに「飛ぶ？」という命令を定義し、殆どの子オブジェクトは継承元の「鳥」の命令を参照するが、「ペンギン」は例外として「飛ぶ？」を上書きする例である。この構造を図示したものを図 8 に示す。

2 回目の演習で用いたプログラムを図 9 に示す。「動物」という基底のオブジェクトに「鳴き声？」という命令を定義し、「鳴き声？」の中では「自分:声」を参照している。「犬」と「猫」がそれぞれ子オブジェクトとして作られ、それぞれに「声」というプロパティが定義されている。この構造を図示したものを図 10 に示す。

また、自分でイメージしたモデルを自分でプログラムの形に組み上げられるのかを測ることでルールの妥当性については検証できると考えたため、それぞれの演習のあとに「演習のプログラムの形で表現できるモデルを自分でイメージしてプログラムすること」という課題を出した。

6. 学生が作ったプログラム

図 11 に演習 1 に対する課題について、学生が書いたプログラム例を示す。果物は基本的には甘いという考えのもの

```

1 動物 = 単語！作る。
2 動物:鳴き声？ = 「|A|
3   クローバー！（A+「の鳴き声は"+自分:声+です。」）話す。
4  ♂
5  犬 = 動物！作る。
6  犬:声 = "ワン"。
7  猫 = 動物！作る。
8  猫:声 = "ニャ〜"。

```

図 9 授業内で使用した演習 2 の内容

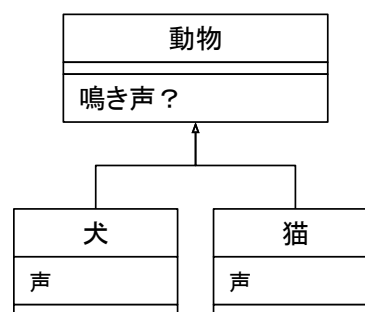


図 10 図 9 の階層構造

と、「果物」という既定オブジェクトに「甘い？」命令を追加し、例外である「レモン」は「甘い？」を上書きしている。

図 12 も同様に、OS は基本的には Linux であるという考えのもと、「OS」という既定オブジェクトに「Linux？」命令を追加し、例外である「Windows」「mac」では「Linux？」を上書きしている。OS は基本的には Linux であるという考えが正しいかは別として、演習内容で得た記述方式やオブジェクトの階層関係を自分で理解し、工夫できていることはわかった。

図 13 に演習 2 に対する課題について、学生が書いたプログラム例を示す。「動物」という基底オブジェクトに「場所？」を定義し、実際の住処の情報はそれぞれ子オブジェクトのプロパティに定義している。図 14 も同様に、「物質」という基底オブジェクトに「材質は？」を定義し、実際の材質の情報はそれぞれ子オブジェクトのプロパティに定義している。

今回検討したルールを基に学生が自分で考えてプログラムの形に落とし込んでいるところから、学生はルールを理解し、自分で工夫できていることがわかる。そのため、今回のルールの妥当性に問題はないのではないかと考えている。

7. まとめ

スマートスピーカーに短文を話しかけた際の応答を記述する記述方式について検討し、提案した。従来だと単語に対する処理の定義が行えなかったが、今回の提案内容により、短文の認識が可能になった。また、大学生を対象とした実験授業により、ルールの妥当性に関しては問題がな

```

1 果物=単語!作る。
2 果物:甘い?=" | A | クローバー !( A+"は甘いです" ) 話す
3 リンゴ=果物!作る。
4 レモン=果物!作る。
5 レモン:甘い?=" | A |
6   クローバー !( A+"は甘くないです" ) 話す
7   。
```

図 11 学生が課題 1 で作ったプログラム例 1

```

1 OS = 単語!作る。
2 OS : Linux ? = " | A |
3   クローバー !( A + "は Linux" ) 話す
4   。
```

```

5
6 Ubuntu = OS !作る。
7 Debian = OS !作る。
8 CentOS = OS !作る。
9
10 Windows = OS !作る。
11 Windows : Linux ? = " | A |
12   クローバー !( A+"は Linux ではない" ) 話す
13   。
```

```

14 mac = OS !作る。
15 mac : Linux ? = " | A |
16   クローバー !( A+"は Linux ではない" ) 話す
17   。
```

図 12 学生が課題 1 で作ったプログラム例 2

```

1 動物=単語!作る。
2 動物:場所?=" | A |
3   クローバー !( A+"の場所は"+自分:住処+"です" ) 話す
4   。
```

```

5 犬=動物!作る。
6 犬:住処="犬小屋"。
7 猫=動物!作る。
8 猫:住処="こたつ"。
9 熊=動物!作る。
10 熊:住処="洞窟"。
```

図 13 学生が課題 2 で作ったプログラム例 1

```

1 物=単語!作る。
2 物:材質?=" | A |
3   クローバー !( A+"の材質は"+自分:質+"です" ) 話す
4   。
```

```

5 コップ=物!作る。
6 コップ:質="プラスチック"。
7 靴=物!作る。
8 靴:質="革"。
```

図 14 学生が課題 2 で作ったプログラム例 2

参考文献

- [1] 本多佑希, 島袋舞子, 浅子秀樹, 兼宗進: スマートスピーカーのアプリケーション開発を支援するプログラミング学習環境の開発, 情報処理学会, 情報教育シンポジウム論文集, Vol.2019, pp.62-68, 2019.
- [2] 大阪電気通信大学 兼宗研究室: プログラミング言語「ドリトル」, 入手先 <http://dolittle.eplang.jp/> (参照 2020-10-15).
- [3] NTT コミュニケーション科学研究所: 日本語語彙体系, 岩波書店 (1997).
- [4] 情報処理学会: カリキュラム標準 J17, 入手先 https://www.ipsj.or.jp/annai/committee/education/j07/curriculum_j17.html (参照 2020-10-15).
- [5] 松澤芳昭, 中鉢欣秀, 岡田健, 大岩元: オブジェクト指向技術者養成のためのカリキュラム, 情報処理学会, コンピュータと教育研究会第 64 回研究発表会, Vol.2002, No.39(2002-CE-064), pp.1-8, 2002.
- [6] 高井久美子, 渡辺博芳, 佐々木茂, 鎌田一雄: 個別学習と協調学習を組み合わせた授業例-オブジェクト指向モデリング導入教育における設計と実践-, 教育システム情報学会誌, Vol.28, No.3, pp.210-222, 2011.

いという結論に至った。

今後はこの記述方式をもとに、中学生や高校生でもこのルールを理解してオブジェクトの階層構造を想像しながらプログラムが組めるのかを検証したい。