

# 知能プログラミング演習 II 課題 2

グループ 8

29114060 後藤 拓也

2019 年 10 月 28 日

■提出物 rep3

■グループ グループ 8

■グループメンバー

学生番号	氏名	貢献度比率
29114003	青山周平	no
29114060	後藤拓也	no
29114116	増田大輝	no
29114142	湯浅範子	no
29119016	小中祐希	no

■自分の役割 必須課題 3.3

「知識システムの質問応答システム」

## 1 課題の説明

課題 3-1 または 3-2 で作った知識表現を用いた質問応答システムを作成せよ。なお、ユーザの質問は英語や日本語のような自然言語が望ましいが、難しければ課題 2 で扱ったような変数を含むパターン (クエリー) でも構わない。

## 2 手法

1. 課題 2 で扱ったような変数を含むクエリーによる質問
2. 英文による質問

### 2.1 手法 1 に関して

変数 (?x や ?y など) を含み, [Tail, Label, Head] の形を守って代入する. 具体的には「Taro hobby baseball」という英文ではなく, 単なるクエリーに対して, 「Taro hobby ?x」といったクエリーの形に添った質問をするといった手法である.

### 2.2 手法 2 に関して

ある一定の知識システムが構築された状態において, 質問する内容というのは自ずと限られてくる. 今回は質問とそれに基づく応答を大きく以下の 2 パターンに分けている.

1. 疑問詞 What に基づく質問から, ?x の内容を答える
2. 質問の内容に対して, Yes か No かで答える

1. の疑問詞を含む質問に関しては, 質問の内容からどのように Head, Tail, Label を取り出すかは, 以下の図 1 を参照してほしい.

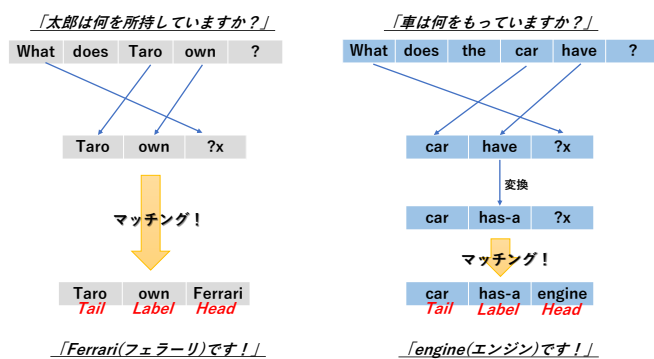


図 1 SVO 構造による疑問詞の質問

英語の第3文型 SVO 型は、S(主語) が Tail, V(動詞) が Lavel, O(目的語) が Head になっている。その形に沿って、疑問文から要素を抽出する。もちろん、前置詞の処理や、have を has-a 関係に合わせるなどの調整も必要である。

この質問は”What を使って問われる内容は、目的語の部分のみ”というのを暗示している。「車はエンジンを持つ」という1文が存在する際に、「車は何を持ちますか?」とは聞けても、「エンジンを持っているのは何ですか」とは聞けない。

2. の Yes か No で答える質問に関しても、以下の図2のように分解できる。

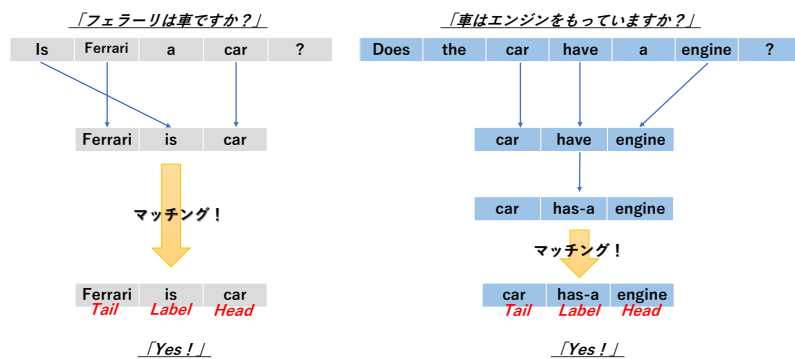


図2 Yes/Noで答える型

同様に SVC をそれぞれ Tail, Lavel, Head に合わせ、マッチングが成功したら、「Yes」を返し、失敗した場合は「No」を返す。

そして、だれもが違和感を覚えたのは、「Taro hobby baseball」という1文。何となく「太郎の趣味は野球です」になるが、Google 翻訳にかけたら、「ヒロキホビーサッカー」である。そもそも hobby は動詞になり得ない。正しくは、「Taro's hobby is baseball」である。これに質問をするには、「Is Taro's hobby a baseball?」に対応する必要がある。そのため、図2のような Label の取り方ではいけない。質問において、2つ目のトークンに「's」があるかどうかで処理を変える。

## 3 実装

### 3.1 手法 1

まずは、手法 1 に関する「課題 2 で扱ったような変数を含むクエリーによる質問」の部分を実装したソースコード 1 に示す。

ソースコード 1 クエリーの形に添った質問応答

---

```
1  /**
2   * 課題 2 で扱ったような変数を含むパターン (クエリー) による質問応答システム
3   * "?x is-a sports" と "?y hobby ?x" をとらえる
4   * → 質問は 3 つのトークンに分けられる
5   */
6  Scanner stdIn1 = new Scanner(System.in); // 文字列読み込み
7  Scanner stdIn2 = new Scanner(System.in); // 数値読み込み
8  ArrayList<ArrayList<String>> queryList
9      = new ArrayList<ArrayList<String>>(); // 質問 (query) を入れる
10 StringTokenizer st; // トークンごとに分解
11 int retry;
12 do {
13     ArrayList<String> tokenList = new ArrayList<>();
14     System.out.println("質問を入力してください");
15     String s = stdIn1.nextLine(); // 質問文がここに入り,
16     st = new StringTokenizer(s); // トークンごとに分解し,
17     for(int i=0; i<st.countTokens(); i++) {
18         tokenList.add(st.nextToken());
19     }
20     tokenList.add(st.nextToken());
21     queryList.add(tokenList);
22     System.out.println("もう 1 つ? 1...Yes/ 0...No");
23     retry = stdIn2.nextInt();
24 }while(retry == 1);
25
26 ArrayList<Link> query = new ArrayList<Link>();
27 for(int i=0; i<queryList.size(); i++) {
28     query.add(new Link(queryList.get(i).get(1), queryList.get(i).get(0),
29         queryList.get(i).get(2)));
29 }
30 sn.query(query);
```

---

上記のプログラムは main 文において、全てのリンクを SemanticNet クラスのインスタンスに加えて後で行われ、その後、query メソッドを呼び出している。質問文は Java の文字列読み込みの Scanner クラスを用いている。それにより入力された String 型のデータ

を空欄があるたびに分割する StringTokenizer を利用して各トークンに分け、それぞれを適切に Tail, Label, Head に分け、query メソッドを呼び出している。

## 3.2 手法 2

次に、「英語の質問応答システム」の条件分岐の部分をソースコード 2 に示す。

ソースコード 2 英語における質問応答

---

```
1      ArrayList<String> tokenList = new ArrayList<>();
2      System.out.println(" 質問を入力してください");
3      String s = stdIn1.nextLine(); // 質問文がここに入り,
4      st = new StringTokenizer(s); // トークンごとに分解し,
5
6      String firstToken = st.nextToken();
7      String secondToken = st.nextToken();
8      if(firstToken.equals("What")) {
9          if(secondToken.equals("does")) {
10             String thirdToken = st.nextToken();
11             if(thirdToken.equals("the"))
12                 thirdToken = st.nextToken();
13             tokenList.add(thirdToken);
14
15             String forthToken = st.nextToken();
16             if(forthToken.equals("have"))
17                 tokenList.add("has-a");
18             else
19                 tokenList.add(forthToken);
20             tokenList.add("?x");
21         }
22
23         else if(secondToken.equals("is")) {
24             String thirdToken = st.nextToken().replace("'s", "");
25             tokenList.add(thirdToken);
26             tokenList.add(st.nextToken());
27             tokenList.add("?x");
28         }
29     }
30     else if(firstToken.equals("Is")) {
31         if(secondToken.contains("'s")) {
32             tokenList.add(secondToken.replace("'s", ""));
33             .
34             . // 以下 "Is" における "Yes, No 返答" の
35             . // 細かい条件分岐が行われている.
```

---

各トークンのメソッドを参照する nextToken メソッドは呼び出すたびに、次のトークンへ参照先が移ってしまうので、条件分岐をする際には、firstToken, secondToken のように一度 String 型に格納している。2 の最初の if 文では、手法 2 で述べた 1 つ目の応答パターン「疑問詞 What に基づく質問」の詳細と、2 つ目の応答パターン「Yes, No で答える質問」の冒頭が示されている。疑問詞 What の後には、SVO 関係の場合には”does”が、SVC 構文の時には”is”とさらに 2 パターンに分かれている。前置詞 the を飛ばしたり、have を has-a に変更するなどを行う。

ここでポイントになるのは、質問文には必ず”(空欄) + ?”を入れてもらいたいということだ。というのも、nextToken メソッドは呼び出し後に次のトークンを参照するので、もし”(空欄) + ?”がなければ、文を最後まで参照した後に、参照する次のトークンがなくなってしまう、エラーになってしまうからである。

## 4 実行例

### 4.1 手法 1

日本語で言うと、[スポーツを趣味にしている人はだれか?] と質問したときの実行結果が以下ようになる。

---

```
1 Successfully started
2 検索結果を取得
3 質問を入力してください
4 ?x is-a sports
5 もう 1 つ? 1...Yes/ 0...No 1
6 質問を入力してください
7 ?y hobby ?x
8 もう 1 つ? 1...Yes/ 0...No 0
9 *** Query ***
10 ?x =is-a=> sports
11 ?y =hobby=> ?x
12 [{?x=baseball, ?y=Taro}]
```

---

まずはスポーツが何か (?x) を求め、その後、そのスポーツに該当する何か (?y) を趣味としている人を探す。正しい関係性が出力されていることが確認される。

ここで注目したいのは、1 回の実行における複数の質問内容は”かつ”の条件で結ばれているということである。これはソースコード 1 をみるとわかることだが、do-while でループさせることで、query メソッドに入れる内容が増え、SemanticNet クラスの queryLink メソッドで、新しいリンクが構築されていくからである。

## 4.2 手法 2

---

```
1 質問を入力してください
2 What is Taro's speciality ?
3 *** Query ***
4 Taro =speciality=> ?x
5 {?x=AI}です.
6 もう 1 回? 1...Yes/ 0...No 1
7 質問を入力してください
8 Is Taro a NIT-student ?
9 *** Query ***
10 Taro =is-a=> NIT-student
11 Yes!
12 もう 1 回? 1...Yes/ 0...No 1
13 質問を入力してください
14 Is Taro a student ?
15 *** Query ***
16 Taro =is-a=> student
17 Yes!
```

---

適切な英語で質問をすることで、結果が返ってくる。Goole 翻訳を使うことをお勧めする。上記の例では、「太郎は NIT の生徒」と「生徒は勉強しない」から「太郎は勉強しない」がうまく結果として出力されていることがわかる。

また、手法 1 とは異なり、“かつの関係”をもって複数の質問をする機能はつけていない。入力した 1 つ質問に対して、1 つの答えが出力される。

## 5 考察

次の質問とそれの回答を見てほしい。

---

```
1 *** Query ***
2 Taro =own=> ?x
3 {?x=Ferrari}です.
4 もう 1 回? 1...Yes/ 0...No 1
5 質問を入力してください
6 Is Ferrari a car ?
7 *** Query ***
8 Ferrari =is-a=> car
9 Yes!
10 もう 1 回? 1...Yes/ 0...No 1
11 質問を入力してください
12 Does Taro own car ?
```

```
13 *** Query ***
14 Taro =own=> car
15 No...
```

---

文章的には、「太郎はフェラーリを持っていて、フェラーリは車なのだから、太郎は車をもっている」と思うが、このプログラムの継承の仕方ではうまくいかない。継承には Label 名が関わっているのである。以下の図を見てもらいたい。

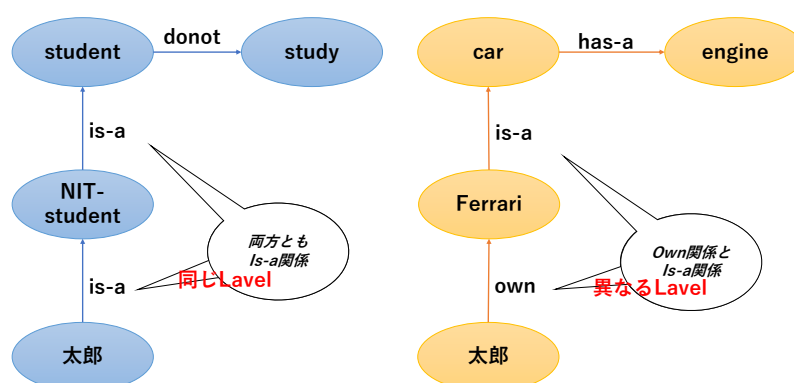


図 3 継承と Label の対応

太郎と Ferrari のラベル”own”と Ferrari と car のラベル”is-a”は異なるため、太郎は car を持っていることにはならない。同様に車はエンジンを持つので、つまり太郎はエンジンをもつということは、文脈上成り立つが、この知識表現ではラベルの違いから成り立たない。その一方で、”太郎は勉強をしない”という文は成り立つ。これは、”is-a”ラベルが NIT-student も student にも成り立ち、太郎は student までつながれるからである。

## 6 感想

Java の使い方は、ググってもいいが、昔しっかり使い込んだ教科書に立ち戻るのも、また一挙である。今回は Scanner クラスと do-while 文の複数入力プログラムを Java 入門の教科書から参照した。

英文での質問を処理する際に、「a」とか「the」とかの前置詞を取るのがめんどくさかった。私はあまり英語が得意ではないので、Google 翻訳先生に頼り、前置詞を補完、処理していった。前置詞は日本人にとってなじみにくい...



## 参考文献

[1] Java による知能プログラミング入門 ー著：新谷 虎松

[2] 新・明解 Java 入門 ー著：柴田望洋

[3] Java 指定型の読み取り ー著：Let's プログラミング

<https://www.javadrive.jp/start/scanner/index2.html>

[4] Google 翻訳 ー著：Google

<https://translate.google.com/?hl=ja>