研究班番号【 107 】

ルールベース機械翻訳によって常体⇔敬体へと変換するプログラム

情報班：小山　翼

１．はじめに

今日、外国人観光客や在留外国人が増え、外国人が日本語を使う機会が増えた。その時、彼らを困らせるのが敬語の使い方である。日本語の敬語はとても複雑であり、日本人でさえ、敬語を正しく使えない人もいる。しかし、日本で暮らすには敬語の使用は避けられない。以上の理由から、常体の文を敬体の文に翻訳するプログラムを作成すれば、日本語が得意でない人にとって、敬語の学習をする時や目上の人と話す時等、様々な場面で助けになるのではないかと考え、この研究を進めたいと思った。機械翻訳には、コーパスと呼ばれる言語データベースを用いて翻訳するルールベース機械翻訳と、大量の対訳をAIに学習させて翻訳する統計的機械翻訳がある。統計的機械翻訳については、敬語の対訳が少なく、プログラムを作成するのが困難と考えた。よってルールベース機械翻訳を用いて研究を進める。

２．作成方法

プログラミング言語はpython、実行環境はGoogle Colaboratoryで行った。形態素解析にはMeCabという形態素解析エンジンを用いた。また、翻訳に必要な辞書はIPA辞書を用いた。

≪方法≫

(1)pythonの辞書型を用いてよく使われる尊敬語の表(sonkeigo)と謙譲語の表(kenjogo)を作成した(図1参照)。この表を用いて単語の変換を行う。

(2)MeCabを用いて、翻訳したい文について形態素解析をする関数(mecab)を作成した(図2参照)。この関数では翻訳したい文を引数(task)として、原文を単語に区切ったリスト(sentence)、それぞれの単語の原型のリスト(genkei)、それぞれの単語の活用形のリスト(katuyo)、それぞれの単語の品詞のリスト(hinshi)の 4 つの戻り値を返すようにした。また、尊敬語と謙譲語の両方に訳せるようにするため、変数sonkeiが１ならば尊敬表現に、変数kenjoが１ならば謙譲表現に翻訳するようにした。

(3)genkeiの中にsonkeigoもしくはkenjogoと共通した単語を探し、その単語に対応する敬語に変換するプログラムを作成した(図 3 参照)。しかし、変換先の敬語を正しく活用させなければ不自然な文になってしまうことが判明した。例えば、「食べた」の語幹である「食べ」を、そのまま「召し上がる」に変換すると、「召し上がるた」になってしまう。よって、目的単語の原型と原文における目的単語の品詞と活用形から、目的単語を指定された活用形に変換するプログラム(kensaku)を作成した(図4参照)。この関数では、IPA辞書の動詞をまとめたファイル(Verb.csv)を読み込み、その中から入力した単語の原型と活用形に該当する単語を探し出し、戻り値として返すようにした。しかし、変換する単語とその敬語の活用形が一致しない場合があった。例えば、「食べた」における「食べる」の活用形は連用形であるが、「召し上がった」

における「召し上がる」の活用形は連用タ接続である。もし、変換先の敬語の活用形を連用形のまま訳してしまうと、「召し上がりた」になってしまう。そこで、変換する単語に続く助詞が「う」もしくは「た」、「て」であるか判定し、「う」であれば活用形を「〇〇ウ接続」に、「た」「て」であれば「〇〇タ接続」にする関数(henkan)を作成した(図 5 参照)。関数 henkan には関数 kensaku を組み込み、一つの処理にまとめた。

(4)丁寧語を付け加える処理をするため、(3)までの処理をした文を変数 semifinal に格納し、それを関数 mecab の引数として形態素解析を行い、(2)と同じように、単語のリストを sentence に、原型のリストを genkei に、活用形のリストを katuyo に、品詞のリストを hinshi に代入するようにした(図 6 参照)。新しく得たこれらのデータを用いて、丁寧語を付け加える処理を行う。

(5)文中の助動詞「だ」を「です」に変換するプログラムを作成した(図 7 参照)。(3)で作成した関数 henkan を用いて、活用形を合わせて変換できるようにした。

(6)助動詞「ます」を文末に付け加えるプログラムを作成した(図 8 参照)。このプログラムでは、助動詞のリスト(jodoshi)(図 9 参照)を用いて文末の単語が、「ます」が接続できる動詞型の活用をする助動詞であるか、または動詞であるかを判定し、その単語を a とする。そして「ます」は連用形に接続するので、a を連用形にして、「ます」を a の活用形に合わせた上で、a の直後に活用させた「ます」を付け加えるようにした。また、原文に「ます」が含まれている場合は、break を用いてすぐに処理を終えるようにした。

(7)敬体の文を常体の文に翻訳するプログラムも作成するために、(1)で作成した辞書型の尊敬語と謙譲語の表の key と value を逆にした新しい表を作成した(図 10 参照)。これを用いて(1)から(3)の処理を行えば、敬体の文を常体の文にすることができた。また、丁寧語の処理については、助動詞の「です」を「だ」に変換し、助動詞の「ます」を取り除くプログラムを作成した(図 11 参照)。

３．成果

ルールベース機械翻訳によって、特定の動詞を尊敬語もしくは謙譲語に変換した上で丁寧語を付け加える、またはその逆方向に変換するプログラムを作成できた。

例　・「先生はリンゴを食べた」→「先生はリンゴを召し上がりました」

・「私はコロッケを召し上がりました」→「私はコロッケを食べた」

しかし、「聞く」のように「伺う」「拝聴する」といった複数の敬語表現がある単語について、訳し分けができなかった。

４．今後の展望

常体から敬体、敬体から常体の双方向に翻訳するプログラムを作成できた。しかし、ニュアンスの違いによる訳し分けはできなかった。そこで、常体と敬体の対訳データを集め、統計的機械翻訳のプログラムを作成すれば訳し分けも可能ではないか、と考察した。また、今回作成したこのプログラムを誰でも利用できるように、Web フレームワークを用いて、Google Colaboratory においてだけでなく、Google などのブラウザ上でも利用できるようにしたい。

５．謝辞

本研究を進めるに当たり、適切な助言を賜り、丁寧に指導して下さった大阪工業大学の小林裕之教授に深く感謝申し上げます。

６．参考文献ならびに参考 Web ページ

＠ menon(2019-11-08), 『Python と MeCab で形態素解析(on Windows)』,<https://qiita.com/menon/items/f041b7c46543f38f78f7 >,(参照 2019-8-14)

＠R-Yoshi(2018-12-05), 『【機械学習】Google 翻訳(みたいなもの)を自作してみた。』,<https://qiita.com/R-Yoshi/items/9a809c0a03e02874fabb>,(参照 2019-11-6)

```
sonkeigo = {'なさる': 'する',
            'いらっしゃる': '来る',
            'ご覧になる': '見る',
            'お耳に入る': '聞く',
            'おっしゃる': '言う',
            'お思いになる': '思う',
            'ご存じです': '知る',
            'くださる': 'あげる',
            'お受けになる': 'もらう',
            '召し上がる': '食べる',
            'お亡くなりになる': '死ぬ',
            'お気に召す': '気に入る'}


kenjogo = {'おる': 'いる',
           '参る': '来る',
           '拝見する': '見る',
           'ご覧に入れる': '見せる',
           '伺う': '聞く', '申し上げる':
           '言う', '存じ上げる': '知る',
           '差し上げる': 'あげる',
           '頂戴する': '貰う',
           'いただく': '食べる',
           '亡くなる': '死ぬ',
           'お目にかかる': '会う',
           '拝借する': '借りる'}
```

図 1　尊敬語と謙譲語の表

```
def mecab(task):
    tagger = MeCab.Tagger("-Ochasen")
    node = tagger.parseToNode(task)
    sentence = []
    genkei = []
    katuyo = []
    hinshi = []

    while node:
        word = node.surface
        wclass = node.feature.split(",")

        if not wclass[0]=="BOS/EOS":

            sentence.append(word)

            genkei.append(wclass[6])

            katuyo.append(wclass[5])

            hinshi.append(wclass[0])

        node = node.next

    return sentence,genkei,katuyo,hinshi
```

図 2　翻訳する文を形態素解析する関数

```
if sonkei==1:
  #print("尊敬にする")
  num=-1
  for a in genkei:
    num+=1
    if a in sonkeigo:
      katuyokei=katuyo[num]
      keigo=sonkeigo[a]
      hinsi=hinshi[num]

      result=henkan(keigo,hinsi,katuyokei,num,sentence)

      sentence[num]=result



if kenjo==1:
  #print("謙譲にする")
  num=-1
  for a in genkei:
    num+=1
    if a in kenjogo:
      katuyokei=katuyo[num]
      keigo=kenjogo[a]
      hinsi=hinshi[num]

      result=henkan(keigo,hinsi,katuyokei,num,sentence)

      sentence[num]=result

semifinal="".join(sentence)
```

図 3　単語を尊敬語または謙譲語に変換する関数

```
def kensaku(part, word, form):
    if part == "動詞":
      file_name = "Verb.csv"
    elif part == "助動詞":
      file_name = "Auxil.csv"
    else :
      sys.exit()

    f = open(file_name,'r')
    dataReader = csv.reader(f)
    for row in dataReader:
       if word == row[10]:
            if form in row[9]:
                return row[0]
```

図 4　指定された活用形の動詞を検索する関数

```
85 def henkan(keigo,hinsi,katuyokei,num,sentence):
86
87         result=mecab(keigo)
88         tango=result[0]
89         a=tango[-1]
90
91         new_result=kensaku(hinsi,a,katuyokei)
92
93         if len(sentence)>num+1:
94             if sentence[num+1]=="う":
95                 if "ウ接続" in katuyokei:
96                     if new_result==None:
97                       katuyokei=katuyokei.strip("ウ接続")
98
99                 else:
100                  katuyokei=katuyokei.strip("形")+"ウ接続"
101                  if kensaku(hinsi,a,katuyokei)==None:
102                    katuyokei=katuyokei.strip("ウ接続")
103
104             if sentence[num+1]=="た" or sentence[num+1]=="て":
105                 if "タ接続" in katuyokei:
106                     if new_result==None:
107                       katuyokei=katuyokei.strip("タ接続")
108                 else:
109                    katuyokei=katuyokei.strip("形")+"タ接続"
110
111                    if kensaku(hinsi,a,katuyokei)==None:
112                     katuyokei=katuyokei.strip("タ接続")
113
114
115         final_result=(kensaku(hinsi,a,katuyokei))
116
117         if not final_result==None:
118           tango[-1]=final_result
119
120         semifinal=""
121         for a in tango:
122           semifinal+=a
123
124         return semifinal
```

図５　活用形をタ接続もしくはウ接続に変換する関数

```
167 semifinal="".join(sentence)
168 print("↓")
169 print(semifinal)
170
171 #ここから丁寧語
172 result=mecab(semifinal)
173 sentence=result[0]
174 genkei=result[1]
175 katuyo=result[2]
176 hinshi=result[3]
```

図６　（３）までの処理を行った文を再び形態素解析するプログラム

```
178 num=-1
179 for a in genkei:
180   num+=1
181   if a=="だ":
182     if hinshi[num]=="助動詞":
183
184       katuyokei=katuyo[num]
185
186       result=henkan("です","助動詞",katuyokei,num,sentence)
187       sentence[num]=result
188
```

図７　助動詞の「だ」を「です」に変換するプログラム

```
num=len(genkei)
for a in reversed(genkei):
  num-=1
  if a=="ます":
    num-=1

  if genkei[num] in jodoshi:
    result=kensaku("動詞",a,"連用形")
    sentence[num]=result

    katuyokei=katuyo[num]
    result=henkan("ます","助動詞",katuyokei,num,sentence)
    sentence.insert(num+1,result)
    break

  if hinshi[num]=="動詞":
    result=kensaku("動詞",a,"連用形")
    sentence[num]=result

    katuyokei=katuyo[num]
    result=henkan("ます","助動詞",katuyokei,num,sentence)
    sentence.insert(num+1,result)
    break
```

図 8　助動詞「ます」を文末に付け加えるプログラム

```
jodoshi=["せる","させる","れる","られる","がる"]
```

図 9　動詞型の活用をする助動詞のリスト

```
sonkeigo = {'なさる': 'する',
            'いらっしゃる': '来る',
            'ご覧になる': '見る',
            'お耳に入る': '聞く',
            'おっしゃる': '言う',
            'お思いになる': '思う',
            'ご存じです': '知る',
            'くださる': 'あげる',
            'お受けになる': 'もらう',
            '召し上がる': '食べる',
            'お亡くなりになる': '死ぬ',
            'お気に召す': '気に入る'}

kenjogo = {'おる': 'いる',
           '参る': '来る',
           '拝見する': '見る',
           'ご覧に入れる': '見せる',
           '伺う': '聞く',
           '申し上げる': '言う',
           '存じ上げる': '知る',
           '差し上げる': 'あげる',
           '頂戴する': '貰う',
           'いただく': '食べる',
           '亡くなる': '死ぬ',
           'お目にかかる': '会う',
           '拝借する': '借りる'}
```

図 10　尊敬語と謙譲語の表

```
result=mecab(semifinal)

sentence=result[0]
genkei=result[1]
katuyo=result[2]
hinshi=result[3]

for a in reversed(genkei):

  if a=="ます":
    num=genkei.index(a)
    katuyokei=katuyo[num]
    tango=genkei[num-1]
    hinsi=hinshi[num-1]
    result=henkan(tango,hinsi,katuyokei,num,sentence)
    sentence[num-1]=result
    sentence.pop(num)

  if a=="です":
    num=genkei.index(a)
    katuyokei=katuyo[num]
    result=henkan("だ","助動詞",katuyokei,num,sentence)
    sentence[num]=result
```

図 11　丁寧語を省くプログラム