●例文を基にした文章の生成●

文章は、いくつかの単語が順に並ぶことで生み出されます。

猿　も　木　から　落ちる　。

今回は次のような例文をもとに、日本語として意味の通る別の文章を自動的に生成することを目指します。

富山県はブラックラーメンが有名です。石川県は人が減少中。その高校はブラック校則が有名ですか。

ステップ１　「分かち書き」による例文からの単語の抽出

日本語は英語と異なり、語句の間をスペースで区切ったりしないので、どこまでが単語なのかがわかりにくい。

すももももももものうち

↓　これを、単語ごとにスペースで区切ることを「分かち書き」という。

すもも　も　もも　も　もも　の　うち

Python では、**Mecab** や **janome** などのモジュールをインストールすることによって日本語の文章の分かち書きを（多少の不完全さはあるにせよ）自動で行うことができる。（詳細は後ほど。）

例文を **janome** をつかって分かち書きしてみます。

```
富山　県　は　ブラック　ラーメン　が　有名　です　。
石川　県　は　人　が　減少　中　。
その　高校　は　ブラック　校則　が　有名　です　か　。
```

すると、３つの例文は、１７種類の単語で構成されていることがわかります。（**gosu = 17**）

富山　県　は　ブラック　ラーメン　が　有名　です　。　石川　人　減少　中　その　高校　校則　か

これらの語からいくつか選び出して適当に並べてみることで、新しい文章の生成を試みる。と…

ブラック　が　県　校則　人　。　その

有名　石川　は　です　ラーメン。

…これだと「カタコトの日本語」とも言えないような文章になってしまいますね。さて。どうしたものか。

【実習①】Anaconda Prompt を用いて、**janome** をインストールします。

```
> pip install janome
```

すると、spyder で次のように書くことで分かち書きができる。

```
1  # -*- coding: utf-8 -*-
2  from janome.tokenizer import Tokenizer
3  t = Tokenizer()
4  txt_data = '富山県はブラックラーメンが有名です。石川県は人が減少中。その高校はブラック
   校則が有名ですか。'
5  print("*******原文*******")
6  print(txt_data)
7  print("*******分かち書き*******")
8  Word_list = list(t.tokenize(txt_data,wakati=True))
9  print(Word_list)
```

ステップ２　マルコフ連鎖についての基礎知識

話は変わりますが、数学に「マルコフ連鎖」という考え方があります。名前からして難しそうですが、実は単純な話で、**次に起こる出来事が、「今の状態」から確率分岐する**（過去によらない）という考え方です。

数学で習ったさいころの独立試行に近いものがあります。例えば、１つのさいころを繰り返し投げるような「実験」を考えてみると、「過去に１の目がたくさん出たから、次にさいころを振ると１の目は出にくい」ということは**ありません。**さいころが特に細工のない普通のもので、１が出る確率が１／６なら、そのさいころで過去１がたくさん出たか否かに関わらず、次にさいころをふれば１／６の確率で１が出ます。

今この「実験」を、[今の状態:[次にとりうる状態]]という記号で、次のように表現することにします。

```
[1:[1,2,3,4,5,6]
 2:[1,2,3,4,5,6]
 3:[1,2,3,4,5,6]
 4:[1,2,3,4,5,6]
 5:[1,2,3,4,5,6]
 6:[1,2,3,4,5,6]]
```

さいころの場合は１～６の６つの状態をとり、各数字が出る割合が等しいので上記のようになります。

ちょっと一般化してみましょう。たとえば、A,B,C の３つの状態があり、いま A の状態にあるものが状態 B に移る確率が１／４、状態 C に移る確率が３／４であるときは、次のように表現することができます。

```
[A:[B,C,C,C]]
```

つまり、[B,C,C,C]のなかからランダムでどれか一つが選ばれることで、どの遷移が起こるのかが確率的に選択される（状態 C への遷移は、B への遷移の３倍起こりやすい。）ように表現します。

問１：いま状態Ｃにあり、そこから状態Ａに遷移する可能性が２／５、Ｂに遷移する可能性が３／５である。それを記号であらわすとどのように書けるか。空欄を補いましょう。[Ｃ[　　]]

問２：

[A:[B,C,C,C]]

[B:[A,C]]

[C:[A,B,B]]

であるとき、

いま状態 A にあるものが状態 A→状態 B→状態 A→状態 C の順に遷移する確率はいくらか。（分数で回答可。）

ステップ３　例文からのモデルの作成と、マルコフ連鎖による作文

例文をうまく解釈し、そこにマルコフ連鎖の考え方を適用するだけで、それなりに意味の通る日本語の文章を生成することができます。

つまり、先に挙げた１７種類の語それぞれの後に、どの語が続く可能性があるかを例文から整理します。この作業を「model の作成」と呼ぶことにします。（すごく大事な作業です。）

たとえば、例文を分析してみると、「ブラック」という語の後には「校則」または「ラーメン」が続きます。「は」という言葉のあとには１／３の確率で「人」、２／３の確率で「ブラック」が続きます。

１７種類の語の後に続く言葉を整理した（つまり、３つの例文から作られた）モデルは次のようになります。

（BoS は、Beginning of Sentence:文章の開始を意味します。）

作成されたモデル	
'BoS': ['富山', '石川', 'その'], '富山': ['県'], '県': ['は', 'は'], 'は': ['ブラック', '人', 'ブラック'], 'ブラック': ['ラーメン', '校則'], 'ラーメン': ['が'], 'が': [　(a)　], '有名': ['です', 'です'], 'です': ['。', 'か'],	'石川': ['県'], '人': ['が'], '減少': ['中'], '中': ['。'], 'その': ['高校'], '高校': ['は'], '校則': ['が'], 'か': ['。']

つまり、「富山」、「石川」、「その」の３語から１語を選ぶところから文章生成がスタートし、もし「富山」が選ばれたらその後には必ず「県」が続く（「現象」とか「校則」は富山の後には選択されない。）「県」のあとは必ず（２／２の確率で）「は」、「は」という言葉のあとには先ほど示したように１／３の確率で「人」、２／３の確率で「ブラック」…という風にマルコフ連鎖を続けていくことで、意味のある語が選ばれて、「。」が選択されたら終了とすることで文章生成（作文）を行うことが可能…なのではないかということになります。

問３：　(a)　にあてはまる語を答えなさい。ただし、同じ語が複数回入ることがある。

問４：このモデルを使って文章を生成した場合に、３つの例文以外で生成される可能性のある文章を１つ書きなさい。

問５：「石川県はブラック校則が減少中です。」という文章が生成される確率はいくらか。（分数で回答可。）

分かち書きをした 1〜9 行目までのプログラミングに続く形で、たとえば次のようにコードを作成することで、マルコフ連鎖による文章生成ができます。

```
#*******重複のない word_list の作成*********
Word_list_single = []
for word in Word_list:
    if word not in Word_list_single:
        Word_list_single.append(word)
gosu = len(Word_list_single)#重複しない単語の数
print(Word_list_single,gosu,"語")
#*******model の生成*********************
Word_list.insert(0, "。")
Model = [[]for i in range(gosu)]
for i in range(gosu):
    Model[i].append(Word_list_single[i])
print("*******model の生成*******")
print("0 番目の要素に続く可能性のある語を 1 番目、2 番目…の要素に追記します")
#print(model)
for j in range(gosu):
    word = Word_list_single[j]
    for i in range(len(Word_list)-1):
        if Word_list[i] == word:
            Model[j].append(Word_list[i+1])
print(Model)
#*******マルコフ連鎖による作文**************
print("*******マルコフ連鎖*******")
import random
sentence = ""
select_word = "。"#BoS（文章の始まり）の代わりに読点を使います。
for j in range(40):#最大で 40 語の文章を生成
    for i in range(gosu):
        if Model[i][0] == select_word:
            num = random.randint(1,len(Model[i])-1)
            select_word = Model[i][num]#後に続く語をランダムに選択
            break
    sentence = sentence + select_word
    print(Model[i][1:len(Model[i])]) #作文過程を表示する記述。省略可。
    print(sentence) #作文過程を表示する記述。省略可。
    if select_word == "。":#「。」が来ると作文を停止
        break
```

【実習②】実行できることが確認出来たら、4 行目の txt_data をさまざまな文章に変えて実行してみましょう。作家の作品の一部とすることで、その作家風の作文をする…なんてこともできるかもしれませんね。