人工無能を作ろう！

はじめに

この文章は、筆者が最近ネットでよく見る記事を個人的に分かりやすく整理してみた結果生み出された駄文です。心してお読みください。

皆さんは「人工無能」というものを知っているでしょうか。人工知能と対比されていそうな名前ですが、歴とした人工知能の一部で、主に会話を行うものを人工無能、またはチャットボットと呼んでいます。

そんな人工無能ないしチャットボットですが、似たようなものを比較的に簡単に作る方法があります。それがマルコフ連鎖を使うことです。「マルコフ連鎖とは何や？」と思っている方は安心してください。後で書きます。

この文章ではマルコフ連鎖を使った簡単な文章生成を通じて、マルコフ連鎖とは何か、チャットボットとは何かを少しでも読者の皆さんに伝えることができれば幸いです。想定している読者は以下の通りです。

- Twitter とかでよく見る文章自動生成[*1]を行いたい人。
- 友達がいないので AI とお話ししたい人。
- ちょっとぽいもの作ってイキリたい人。
- 暇な人。

本文では主に Python のコードを使用して説明しますが、他にも様々な言語で同じことが可能です。Python アレルギーの方はお帰りください。

また注意点として、Python の環境構築や Twitter から文章を持ってきて学習させる方法はここでは書きません。ググれば無限に記事が出てきます。

この文章では、基本的な専門用語も脚注などで軽い説明を入れてあるつもりです。わからないところが出てきても、とりあえず少し先まで読んでみてから調べることをお勧めします。

最後に、本書に関してご意見などありましたら筆者の Twitter(@50m_regent) までご意見をいただければ幸いです。

では、チャットボットの世界に飛び込みましょう！

マルコフ連鎖って何？

まず Wikipedia には何と書いてあるか見てみましょう。

[*1] しゅうまい君とか

> マルコフ連鎖とは、確率過程の一種であるマルコフ過程のうち、とりうる状態が離散的なものをいう。また特に、時間が離散的なものを指すことが多い。

この時点で意味不明ですね。もっと簡単に説明します。
次の二つの文章を見てみましょう。

- 今日もいい天気ですね。
- 明日も悪い天気でしょう。

この文章を単語ごとに分解すると、

- 今日, も, いい, 天気, ですね, 。
- 明日, も, 悪い, 天気, でしょ, う, 。

となります。

この二つの文章だけを見ると、’も’のあとはそれぞれ 50% の確率で’いい’、’悪い’が続くことが分かります。逆に、’今日’と’明日’の後には絶対’も’が続きます。

このように、今の状態[*2]をみた上で、次の単語を選択していくのがマルコフ連鎖です。

また、n 個の単語をみて次の単語を選ぶこと[*3]を n 階マルコフ連鎖と言います。階数を増やせば増やすほど、元の文章の再現度が上がりますが、生成される文章のパターンが減ってしまいます。階数の調節も、マルコフ連鎖による文章生成の面白いところです。

簡単な文章生成をしてみよう

上の節で説明したマルコフ連鎖を使って文章を生成してみたいと思います。Python コードを書きますが、意味を全て日本語で書くので、Python がわからない人は自分の好きな言語で書いてみてください。

一部 Janome というモジュール[*4]を使っていますが、いろんな言語でその代わりになるものがあると思うので、”{ 言語名 } 分かち書き” とかで調べてみてください。

文章の読み込み、辞書への登録

始めに、文章を渡したらそれを元に辞書を作成してくれる関数を書きます。ソースコード 1 を参考にしてみてください。

ソースコード 1　辞書を作成する関数

```
1 # 分かち書きをしてくれるクラス
2 from janome.tokenizer import Tokenizer
3
4 # マルコフ連鎖の階数
5 CHUNK_LENGTH = 2
6
```

[*2] 文章を前からみていくするときに、一番最後に選んだ単語
[*3] $n = 2$ の例: ’今日’と’も’をみて、’今日も’の後に続く’いい’を予想
[*4] 簡単にいうと先人の知恵。いろいろな機能がまとめられてるプログラムのこと。

```
# 文章を受け取って辞書に登録する関数
def make_dict(text):
    # 単語ごとの配列にする。分かち書きの引数を設定すると分かち書きになる
    words = Tokenizer().tokenize(text, wakati=True)

    # 最後に読んだ単語の配列。二個持っておく
    keywords = []
    # 辞書。三個の単語に対してそれに続いたことのある単語たちを登録する。
    markov_dict = {}

    # 文章中の単語を一つずつ見る
    for word in words:
        # に二つ単語が入っていたらそれに続く単語を登録 keywords
        if len(keywords) == CHUNK_LENGTH:
            # 辞書にの単語たちが始めて出現していたら空配列で初期化 keywords
            if tuple(keywords) not in markov_dict:
                markov_dict[tuple(keywords)] = []
            # 次に続く単語を登録
            markov_dict[tuple(keywords)].append(word)

            # 最後に追加した単語を後ろ二つにする
            keywords = keywords[1:]
        # 今出現した単語を追加
        keywords.append(word)

    # 辞書を返す
    return markov_dict
```

入力に先ほどの文を渡してみましょう。

すると、words = [’ 今日’, ’ も’, ’ いい’, ’ 天気’, ’ ですね’, ’。’, ’ 明日’, ’ も’, ’ 悪い’, ’ 天気’, ’ でしょ’, ’ う’, ’。’] となります。

この単語リストから辞書を生成すると、dict = { (’ 今日’, ’ は’): [’ いい’], (’ は’, ’ いい’): [’ 天気’], (’ いい’, ’ 天気’): [’ です’], (’ 天気’, ’ です’): [’ ね’], (’ です’, ’ ね’): [’。’], (’ ね’, ’。’): [’ 明日’], (’。’, ’ 明日’): [’ も’], (’ 明日’, ’ も’): [’ 悪い’], (’ も’, ’ 悪い’): [’ 天気’], (’ 悪い’, ’ 天気’): [’ でしょ’], (’ 天気’, ’ でしょ’): [’ う’], (’ でしょ’, ’ う’): [’。’] } となります。

今回は文章が短いので、辞書も小さいですが、もっと長い文章を入力すると大きい辞書を作ることができます！

また、’。’ を含んだ状態を作らないなどの工夫をするとより性能をあげることができるので是非やってみてください。

辞書から文章生成

次に、作成した辞書から文章を生成する関数を書きます。

ソースコード 2　文章を生成する関数

```
# ランダムに選択する関数
from random import choice

# 受け取った辞書を元に文を生成する関数
def generate_sentence(markov_dict):
    # 文を空文字列で初期化
    sentence = ''
    # 最初の状態をランダムで選択
    keywords = list(choice(list(markov_dict.keys())))

    # 文が終わるまで無限ループ
    while True:
        # 次の単語を辞書の中からランダムで選択して文に追加
        next_word = choice(markov_dict[tuple(keywords)])
        sentence += next_word

        # 今選んだ単語が「。」なら終わり
        if next_word == 。:
            break

        # 状態を後ろ二つにして、今出現した単語を追加
        keywords = keywords[1:] + [next_word]

    # 文を返す
    return sentence
```

8 行目の dict.keys() は、辞書に登録されてる状態の一覧を指します。

dict.keys() = [(' 今日', ' は'), (' は', ' いい'), (' いい', ' 天気'), (' 天気', ' です'), (' です', ' ね'), (' ね', '。'), ('。', ' 明日'), (' 明日', ' も'), (' も', ' 悪い'), (' 悪い', ' 天気'), (' 天気', ' でしょ'), (' でしょ', ' う')]

最初にこれらの状態の中からランダムに一つを選んで文章生成を始めて、'。' が出てきた時点で文章を返します。

プログラムの実行

最後に、これらのコードを実行用のコードをまとめて載せるので、実行してみましょう。

ソースコード 3　全体のコード

```
from janome.tokenizer import Tokenizer
from random import choice

CHUNK_LENGTH = 2

def make_dict(text):
    words = Tokenizer().tokenize(text, wakati=True)
```

```
8
9       keywords = []
10      markov_dict = {}
11
12      for word in words:
13          if len(keywords) == CHUNK_LENGTH:
14              if tuple(keywords) not in markov_dict:
15                  markov_dict[tuple(keywords)] = []
16              markov_dict[tuple(keywords)].append(word)
17
18              keywords = keywords[1:]
19          keywords.append(word)
20
21      return markov_dict
22
23  def generate_sentence(markov_dict):
24      sentence = ''
25      keywords = list(choice(list(markov_dict.keys())))
26
27      while True:
28          next_word = choice(markov_dict[tuple(keywords)])
29          sentence += next_word
30
31          if next_word == 。:
32              break
33
34          keywords = keywords[1:] + [next_word]
35
36      return sentence
37
38  if __name__ == '__main__':
39      # 文章で入力を受け取る
40      text = input文章 ('<< ')
41
42      # 辞書を作成
43         markov_dict = make_dict(text)
44         # 文を生成
45         sentence = generate_sentence(markov_dict)
46
47      # 文を出力
48         print(sentence)
```

今回は検証用に宮沢賢治の「注文の多い料理店」を使用します。生成された文章をいくつか載せます。

- はいっても、もうもとのとおりになおりませんでした。
- さいふやネクタイピンは、西洋料理店というのは、どうも酢のような犬が封筒になって戻ってきて、来

た人を西洋料理を、頭へぱちゃぱちゃ振りかけました。

- 穴が二つつき、銀いろのホークとナイフの形が切りだして、それらがさがさ鳴りました。

どうでしょう。変な日本語ではありますが、それなりの文章が生成されてるのではないでしょうか。文体はまさに宮沢賢治のそれです。マルコフ連鎖の階数を調節したり、他の工夫を施すことでいろいろな文章を生成することができます。

おわりに

初めての文章生成はどうでしたでしょうか。この記事が皆さんの記憶に少しでも残れば幸いです。

人工無能に興味を持った方は、いろいろもっと先を調べてみて、是非会話機能などを作ってみてください。もちろんツイッターなどで聞いてくださってもいいです。

ここまで読んでくださってありがとうございました。良い人工無能ライフを！