HW 1 Code

Jiping Lin A15058075

October 5, 2021

[1]:
```
# CSE 250 HW1
# Jiping Lin A15058075
import string


def load_data():
    word_dic = {}
    with open("hw1_word_counts_05.txt", "r") as f:
        for line in f.readlines():
            line = line.strip().split(" ")
            word_dic[line[0]] = line[1]
    return word_dic


dic = load_data()


def create_prob_table():
    total = 0
    for count in dic.values():
        total += int(count)
    table = {}
    for key in dic.keys():
        table[key] = int(dic[key]) / total
    return table


prior_prob = create_prob_table()


# Print the most frequent/ least frequent words
def print_most(num):
    dic_sorted = sorted(dic.items(), key=lambda x: int(x[1]), reverse=True)
    for i in range(num):
        print(dic_sorted[i][0])
```

```python
def print_least(num):
    dic_sorted = sorted(dic.items(), key=lambda x: int(x[1]), reverse=True)
    for i in range(1, num + 1):
        print(dic_sorted[-i][0])


# return P(W=w)
def prob_w(word: str):
    # Prior Probability table
    return prior_prob[word]


class State:

    def __init__(self, content=None, out=None):
        if out is None:
            out = {}
            for k in range(5):
                out[k] = set()
        if content is None:
            content = [None] * 5

        self.content = content
        self.out = out

    def add_correct(self, letter: str, pos):
        if pos < 0 or pos > 4:
            return
        self.content[pos] = letter

    def add_false(self, letter):
        for i in range(5):
            self.out[i].add(letter)

    # P(E|W=w)
    def prob_ew(self, word: str) -> int:
        word = word.upper()
        appear_set = set()
        for i in range(len(self.content)):
            if self.content[i] is not None:
                appear_set.add(self.content[i])
        for i in range(len(self.content)):
            if self.content[i] is None:
                if word[i] in self.out[i] or word[i] in appear_set:
                    return 0
                else:
                    continue
            if self.content[i] != word[i]:
                return 0
        return 1

    def get_bot(self):
        bot = 0
        for key in dic.keys():
            bot += self.prob_ew(key) * prob_w(key)
        return bot

    def posterior(self, word: str) -> float:
        bot = self.get_bot()
        pe = self.prob_ew(word)
        pw = prob_w(word)
        return pe * pw / bot

    def predictive_first(self, letter: str, word: str) -> int:
        for i in range(len(self.content)):
            if self.content[i] is None:
                if word[i] == letter:
                    return 1
        return 0

    def predictive(self, letter: str):
        prob = 0
        bot = self.get_bot()
        for key in dic.keys():
            pe = self.prob_ew(key)
            pw = prob_w(key)
            prob += self.predictive_first(letter, key) * pe * pw / bot
        return prob

    def get_next_guess(self):
        result = {}
        letters = []
        for let in string.ascii_uppercase:
            letters.append(let)
        for i in range(len(letters)):
            add = letters[i]
            result[add] = self.predictive(add)
        result_sorted = sorted(result.items(), key=lambda x: x[1], reverse=True)
        index = 0
        return result_sorted[index]


def print_guess(correct: list, false: set):
    current = State()
    for i in range(len(correct)):
        if correct[i] is not None:
            current.add_correct(correct[i], i)
    for i in false:
        if i is not None:
            current.add_false(i)
    print(current.get_next_guess())
```

[2]:
```
def main():
    letters = []
    for let in string.ascii_uppercase:
        letters.append(let)
    # 1.9a
    print_most(15)
    print()
    print_least(14)
    # 1.9b
    # First row
    print_guess([None, None, None, None, None], set())
    # Second row
    print_guess([None, None, None, None, None], {'E', 'A'})
    # Third row
    print_guess(['A', None, None, None, 'S'], set())
    # Fourth row
    print_guess(['A', None, None, None, 'S'], {'I'})
    # Fifth row
    print_guess([None, None, 'O', None, None], {'A', 'E', 'M', 'N', 'T'})
    # Sixth row
    print_guess([None, None, None, None, None], {'E', 'O'})
    # Seventh row
    print_guess(['D', None, None, 'I', None], set())
    # Eighth row
    print_guess(['D', None, None, 'I', None], {'A'})
    # Ninth row
    print_guess([None, 'U', None, None, None], {'A', 'E', 'I', 'O', 'S'})


if __name__ == '__main__':
    main()
```

```
THREE
SEVEN
EIGHT
WOULD
ABOUT
THEIR
WHICH
```

```
AFTER
FIRST
FIFTY
OTHER
FORTY
YEARS
THERE
SIXTY

TROUP
OTTIS
MAPCO
CAIXA
BOSAK
YALOM
TOCOR
SERNA
PAXON
NIAID
FOAMY
FABRI
CLEFT
CCAIR
('E', 0.5394172389647948)
('O', 0.5340315651557679)
('E', 0.7715371621621622)
('E', 0.7127008416220354)
('R', 0.7453866259829711)
('I', 0.6365554141009618)
('A', 0.8206845238095241)
('E', 0.7520746887966806)
('Y', 0.6269651101630528)
```

```
[ ]:
```