HW 11 보고서

2013-10035
언어학과 김영민

1.1

1.1 문제 해결 방법

```python
import numpy as np

# corpus size
N = 1410000000

# N-gram frequency
ngram_dict = {}
ngram_dict['1'] = {'하늘은': 3520000,
                   '파랗고': 392000,
                   '단풍잎은': 34600,
                   '빨갛고': 339000,
                   '은행잎은': 24300,
                   '노랗고': 359000}

ngram_dict['2'] = {'하늘은 파랗고': 56100,
                   '파랗고 단풍잎은': 23,
                   '단풍잎은 빨갛고': 160,
                   '빨갛고 은행잎은': 85,
                   '은행잎은 노랗고': 198}

ngram_dict['3'] = {'하늘은 파랗고 단풍잎은': 34,
                   '파랗고 단풍잎은 빨갛고': 0,
                   '단풍잎은 빨갛고 은행잎은': 3,
                   '빨갛고 은행잎은 노랗고': 85}

def ngram_prob(ngram):
    if '<s>' in ngram:
        return ngram_prob(ngram[1:])
    else:
        n = len(ngram)
        if n == 1:
            prob = ngram_dict[str(n)].get(ngram[0], 0) / N
            print('P(%s): %.32f' % (ngram[0], prob))
            return prob
        prob = ngram_dict[str(n)].get(' '.join(ngram), 0) / ngram_dict[str(n-1)].get(' '.join(ngram[:-1]), 0)
        print('P(%s): %.32f' % (' '.join(ngram), prob))
        return prob

def sentence_probability(n, sentence):
    sentence = ['<s>'] * (n-1) + sentence.split(' ')
    prob = 1
    for i in range(0, len(sentence) - n + 1):
        ngram = sentence[i: i + n]
        prob *= ngram_prob(ngram)
    return prob
```

ngram.py 파일에 이와 같은 메소드를 작성해서 n(=1, 2, 3)을 파라미터로 줬을 때 각각 unigram, bigram, trigram 모델 기준으로 문장의 확률을 계산할 수 있도록 함.

1.1 메소드 설명 sentence_probability()

```
def sentence_probability(n, sentence):
    sentence = ['<s>'] * (n-1) + sentence.split(' ')
    prob = 1
    for i in range(0, len(sentence) - n + 1):
        ngram = sentence[i: i + n]
        prob *= ngram_prob(ngram)
    return prob
```

- 문장을 공백문자로 split 해서 list 로 만들고 n값에 따라 적절한 개수의 <s> 문자를 list 의 앞에 삽입
- 전체 문장을 n-gram 단위로 잘라서 각 n-gram 의 확률(ngram_prob())을 구한 뒤 곱해줌
- 출력: 전체 문장 W 의 확률 P(W)

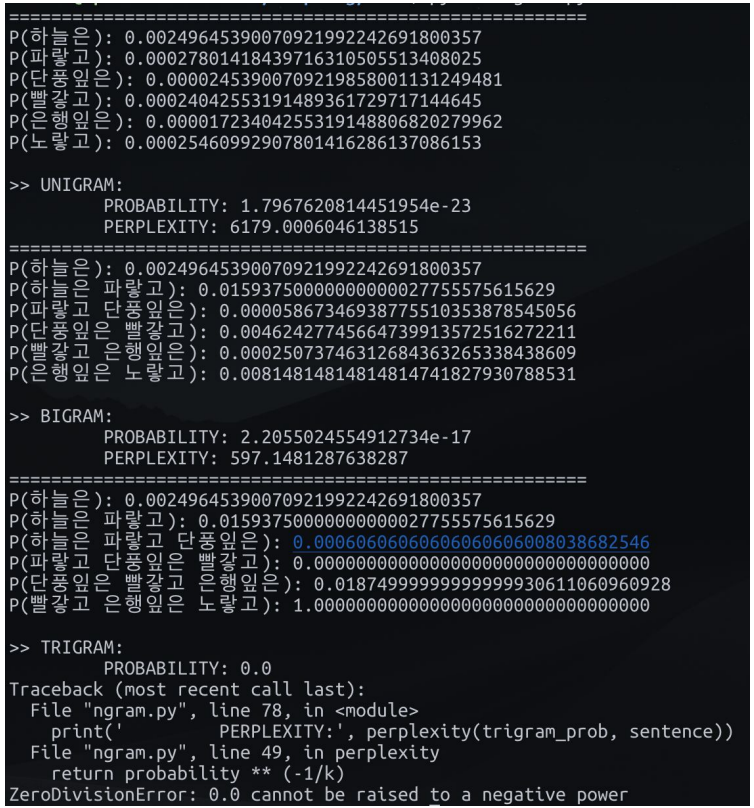
1.1 메소드 설명 ngram_prob()

```
def ngram_prob(ngram):
    if '<s>' in ngram:
        return ngram_prob(ngram[1:])
    else:
        n = len(ngram)
        if n == 1:
            prob = ngram_dict[str(n)].get(ngram[0], 0) / N
            print('P(%s): %.32f' % (ngram[0], prob))
            return prob
        prob = ngram_dict[str(n)].get(' '.join(ngram), 0) / ngram_dict[str(n-1)].get(' '.join(ngram[:-1]), 0)
        print('P(%s): %.32f' % (' '.join(ngram), prob))
        return prob
```

- 문장을 n-gram 단위로 나눴을 때 각 n-gram 의 확률을 구하는 메소드

- 문제에서 <s> 가 포함된 n-gram 은 <s>를 모두 제거한 (n-k)-gram 과 확률이 같다는 가정이 있으므로 <s> 가 있으면 제거하고 확률을 리턴하도록 재귀적으로 처리

1.1 결과



- n= 1, 2, 3, add_k = 1로 코드를 돌린 결과

Unigram: 1.7967620814451954e-23
Bigram: 2.2055024554912734e-17
Trigram: 0.0

- ‘파랗고 단풍잎은 빨갛고’ 의 빈도가 0 이므로 P(W) = 0 의 결과가 나온다.

1.1 결과 검증 (Uni-gram)

```
>>> # UNIGRAM
...
>>> N = 1410000000
>>> p_1 = 3520000 / N # 하늘은
>>> p_1
0.00249645390070922
>>> p_2 = 392000 / N # 파랗고
>>> p_2
0.0002780141843971631
>>> p_3 = 34600 / N # 단풍잎은
>>> p_3
2.453900709219858e-05
>>> p_4 = 339000 / N # 빨갛고
>>> p_4
0.00024042553191489362
>>> p_5 = 24300 / N # 은행잎은
>>> p_5
1.7234042553191488e-05
>>> p_6 = 359000 / N # 노랗고
>>> p_6
0.00025460992907801416
>>> p_1 * p_2 * p_3 * p_4 * p_5 * p_6
1.7967620814451954e-23
```

코드가 의도한대로 돌아가는지 확인해보기 위해 파이썬 shell 로 직접 계산 결과를 구한 뒤 비교해본 결과 메소드를 사용해 구한 값과 일치하는 것을 확인할 수 있었다.

1.1 결과 검증 (Bi-gram)

```
>>> # BIGRAM
...
>>> p_1 = 3520000 / N # <s> 하늘은 == 하늘은
>>> p_1
0.00249645390070922
>>> p_2 = 56100 / 3520000 # 하늘은 파랗고
>>> p_2
0.0159375
>>> p_3 = 23 / 392000 # 파랗고 단풍잎은
>>> p_3
5.8673469387755104e-05
>>> p_4 = 160/ 34600 # 단풍잎은 빨갛고
>>> p_4
0.004624277456647399
>>> p_5 = 85 / 339000 # 빨갛고 은행잎은
>>> p_5
0.00025073746312684363
>>> p_6 = 198 / 24300 # 은행잎은 노랗고
>>> p_6
0.008148148148148147
>>> p_1 * p_2 * p_3 * p_4 * p_5 * p_6
2.2055024554912734e-17
```

1.1 결과 확인 (Tri-gram)

```
>>> # TRIGRAM
...
>>> p_1 = 3520000 / N # <s> <s> 하늘은 == <s> 하늘은 == 하늘은
>>> p_1
0.00249645390070922
>>> p_2 = 56100 / 3520000 # <s> 하늘은 파랗고 == 하늘은 파랗고
>>> p_2
0.0159375
>>> p_3 = 34 / 56100 # 하늘은 파랗고 단풍잎은
>>> p_3
0.0006060606060606061
>>> p_4 = 0 / 23 # 파랗고 단풍잎은 빨갛고
>>> p_4
0.0
>>> p_5 = 3 / 160 # 단풍잎은 빨갛고 은행잎은
>>> p_5
0.01875
>>> p_6 = 85 / 85 # 빨갛고 은행잎은 노랗고
>>> p_6
1.0
>>> p_1 * p_2 * p_3 * p_4 * p_5 * p_6
0.0
```

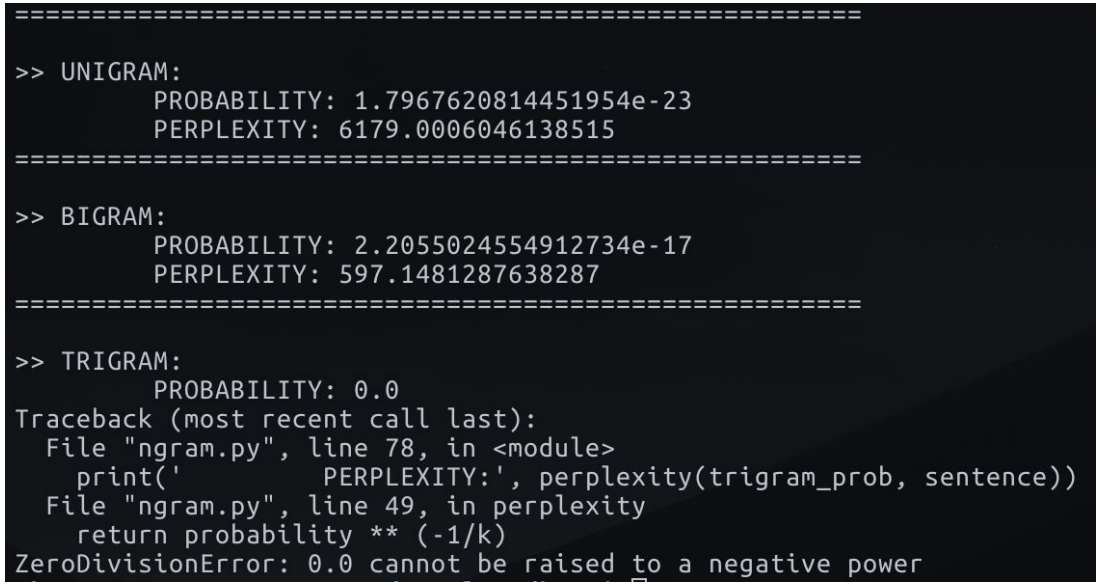
1.2

1.2 메소드 정의

```
def perplexity(probability, sentence):
    k = len(sentence.split(' '))
    return probability ** (-1/k)
```

- 강의 슬라이드에 나온 복잡도 계산식을 그대로 코드로 옮긴 perplexity() 메소드를 작성하였다.
- 문장을 공백 기준으로 split 하여 문장의 길이 (단어수) k를 구한 뒤 각 모델을 통해 구한 문장 전체의 확률에 -1/k 승 한다.

1.2 결과



- 1.1 에서 구한 확률을 perplexity() 함수에 넣어 얻은 복잡도는 이와 같다.
- unigram 보다 bigram 이 확률이 높기 때문에 bigram의 복잡도가 더 낮은 것을 확인할 수 있다.
- trigram은 확률이 0.0 이기 때문에 0.0^(-⅙) 을 계산할 때 에러가 발생한다. (division by zero)

1.3

1.3 메소드 정의

```
def trigram_interpolation(sentence, weights):
    print('weights: %s\n' % str(weights))
    sentence = ['<s>'] * 2 + sentence.split(' ')
    prob = 1
    for i in range(0, len(sentence) - 2):
        tri, bi, uni = sentence[i:i+3], sentence[i+1:i+3], sentence[i+2:i+3]
        model_probs = ngram_prob(tri), ngram_prob(bi), ngram_prob(uni)
        print('weighted probability of %s: %.32f\n' % (" ".join(tri), np.sum([p*w for p, w in zip(model_probs, weights)])))
        prob *= np.sum([p*w for p, w in zip(model_probs, weights)])
    return prob
```

- weights (w_tri, w_bi, w_uni) 가 주어졌을 때 이 값들을 기준으로 보간법을 적용한 확률을 구하는 메소드를 이와 같이 구현하였다.

- 1.1 을 풀 때 사용했던 k-smoothing 기법은 사용하지 않았다. (k 값을 0 으로 설정)

1.3 결과 확인

```
weights: (0.5, 0.3, 0.2)

P(하늘은): 0.0024964539007092199224269180035 7
P(하늘은): 0.0024964539007092199224269180035 7
P(하늘은): 0.0024964539007092199224269180035 7
weighted probability of <s> <s> 하늘은: 0.0024964539007092199224269180035 7

P(하늘은 파랗고): 0.01593750000000000027755575615629
P(하늘은 파랗고): 0.01593750000000000027755575615629
P(파랗고): 0.00027801418439716310505513408025
weighted probability of <s> 하늘은 파랗고: 0.012805602836879433070738087963 04

P(하늘은 파랗고 단풍잎은): 0.000606060606060606060608038682546
P(파랗고 단풍잎은): 0.000058673469387755103538785450 56
P(단풍잎은): 0.000024539007092198580011312494 81
weighted probability of 하늘은 파랗고 단풍잎은: 0.000325540145265069264229190748 59

P(파랗고 단풍잎은 빨갛고): 0.000000000000000000000000000000
P(단풍잎은 빨갛고): 0.004624277456647399135725162722 11
P(빨갛고): 0.000240425531914893617297171446 45
weighted probability of 파랗고 단풍잎은 빨갛고: 0.001435368343377198437071928793 78

P(단풍잎은 빨갛고 은행잎은): 0.018749999999999999930611060960928
P(빨갛고 은행잎은): 0.000250737463126843632653384386 09
P(은행잎은): 0.000017234042553191488068202799 62
weighted probability of 단풍잎은 빨갛고 은행잎은: 0.009453668047448690950340655092 53

P(빨갛고 은행잎은 노랗고): 1.000000000000000000000000000000
P(은행잎은 노랗고): 0.008148148148148147418279307885 31
P(노랗고): 0.000254609929078014162861370861 53
weighted probability of 빨갛고 은행잎은 노랗고: 0.502495366430260048673517303541 30


>> TRIGRAM INTERPOLATION:
        PROBABILITY: 7.096168276562839e-14
        PERPLEXITY: 155.41610326068232
==========================================================
```

```
weights: (0.7, 0.2, 0.1)

P(하늘은): 0.0024964539007092199224269180035 7
P(하늘은): 0.0024964539007092199224269180035 7
P(하늘은): 0.0024964539007092199224269180035 7
weighted probability of <s> <s> 하늘은: 0.0024964539007092194887460490093 6

P(하늘은 파랗고): 0.01593750000000000027755575615629
P(하늘은 파랗고): 0.01593750000000000027755575615629
P(파랗고): 0.00027801418439716310505513408025
weighted probability of <s> 하늘은 파랗고: 0.014371551418439714939423446082 86

P(하늘은 파랗고 단풍잎은): 0.000606060606060606060608038682546
P(파랗고 단풍잎은): 0.000058673469387755103538785450 56
P(단풍잎은): 0.000024539007092198580011312494 81
weighted probability of 하늘은 파랗고 단풍잎은: 0.000438431018829195070281995461 06

P(파랗고 단풍잎은 빨갛고): 0.000000000000000000000000000000
P(단풍잎은 빨갛고): 0.004624277456647399135725162722 11
P(빨갛고): 0.000240425531914893617297171446 45
weighted probability of 파랗고 단풍잎은 빨갛고: 0.000948898044520969262058396331 84

P(단풍잎은 빨갛고 은행잎은): 0.018749999999999999930611060960928
P(빨갛고 은행잎은): 0.000250737463126843632653384386 09
P(은행잎은): 0.000017234042553191488068202799 62
weighted probability of 단풍잎은 빨갛고 은행잎은: 0.013176870896880686287455652916 38

P(빨갛고 은행잎은 노랗고): 1.000000000000000000000000000000
P(은행잎은 노랗고): 0.008148148148148147418279307885 31
P(노랗고): 0.000254609929078014162861370861 53
weighted probability of 빨갛고 은행잎은 노랗고: 0.701655090622537391809032669698 35


>> TRIGRAM INTERPOLATION:
        PROBABILITY: 1.3800156119401195e-13
        PERPLEXITY: 139.10817939561127
```

1.3 결과 확인

- (0.5, 0.3, 0.2)
  - 확률: 7.096168276562839e-14
  - 복잡도: 155.41610326068232
- (0.7, 0.2, 0.1)
  - 확률: 1.3800156119401195e-13
  - 복잡도: 139.10817939561127

- 세 모델 중 trigram 기준 확률이 항상 가장 높기 때문에 trigram 에 더 큰 가중치를 줄수록 확률 또한 높아지는 것을 확인할 수 있다.