

빅데이터 최신기술

문장 생성 확률 계산

소프트웨어학부

20163162

차운성

1. 과제 수행 내용 설명

- 데이터 전처리 과정

1. 먼저 get-gram 프로그램에 -2 옵션을 주어 대용량 말뭉치 파일(corpus.txt)로부터 bigram.txt 파일을 생성한다.
2. 생성된 bigram.txt 파일로 wordcount를 수행하는데, wordcount로 한 번에 실행할 수 없는 경우 split 프로그램으로 파일 분할 후 wordcount를 수행한다

```
$ > split.exe -4m bigram.txt
$ > wordcount.exe -i -new -l xaa
$ > wordcount.exe -i -add -l xab
$ > wordcount.exe -i -add -l xac
```

...

3. wordcount로 생성된 out.txt를 이용해 문장 생성 확률을 계산한다.

- 프로그램 개요

```
$ > calculate-sentence-percentage -freq out.txt
```

freq: 해당 옵션이 없을 경우 default값은 2가 되고, 옵션을 부여할 경우 2보다 높은 값으로 부여한다.

기존 ngram-prob.c를 확장한 프로그램으로, gets()함수를 이용해 문장을 입력받고, 입력된 문장에서 ASCII를 고려하고 2글자씩 끊어서 확률값을 계산한다. 입력받은 문장에 대한 bigram 음절 확률 계산이 끝나면 문장 생성 확률을 출력한다. bigram 음절과 해당 음절의 확률은 구조체에 저장되어있는데, 확률을 구하려는 bigram이 구조체에 없으면, 첫 음절로부터 생성될 수 있는 음절 중 가장 낮은 확률을 가지는 값으로 확률을 구한다. 만약 첫 음절로부터 생성되는 음절이 없는 경우에는 DBL_MIN 값을 곱해주는 방식으로 정의하였다.

키보드로부터 문장을 입력받다가 키보드 입력을 종료하기 위해 q를 입력하면 dic1과 dic2로부터 생성되는 모든 문장에 대해서 확률을 구한 후, 각 케이스별로 높은 확률을 가지는 문장부터 출력하고 프로그램이 종료된다.

- 세부 구조

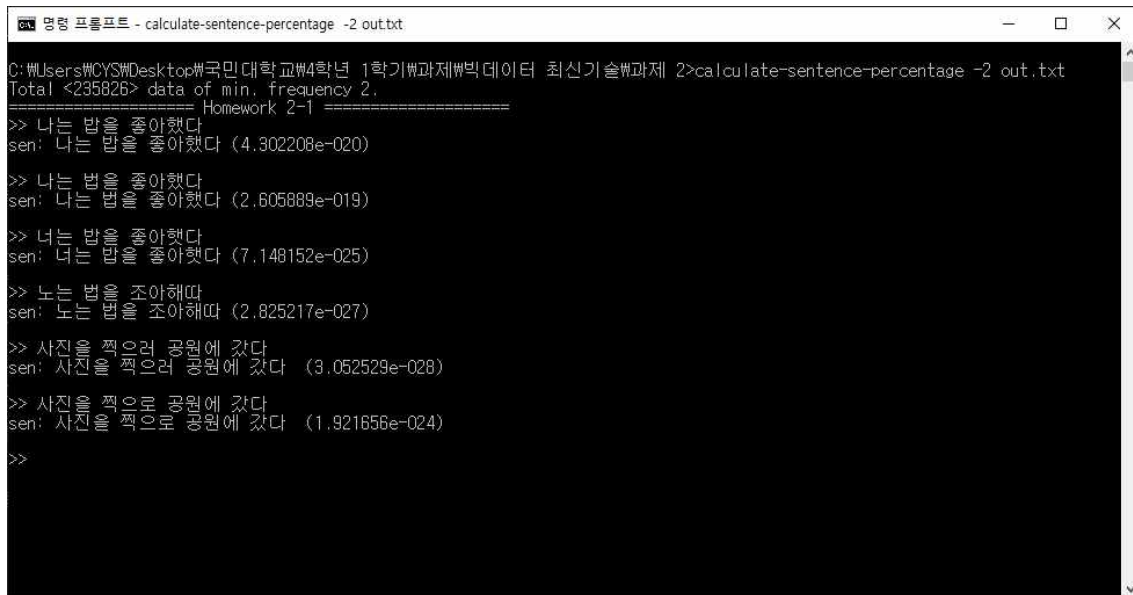
기본 구조는 ngram-prob.c를 그대로 가져왔고, dic1과 dic2에서 조합된 문장과 각 문장의 생성 확률을 가지는 구조체를 정의하고 dic1과 dic2에 대한 구조체 배열을 각각 정의해주었다. 해당 구조체 배열을 Percentage.percentage를 기준 내림차순 sorting을 해주고 구조체 배열을 print 해주면 조합된 문장들의 문장 생성 확률이 높은 순으로 출력된다.

키보드로부터 입력받은 문장과 dic1과 dic2를 통해 조합된 문장들 모두 normalize 과정을 거쳐 ASCII 값들도 2byte로 바꿔준 후 생성 확률을 계산하도록 하였다.

각 음절의 생성확률을 계속해서 곱해나가면 확률값이 상당히 작아지기 때문에 %f 포맷으로 확률값을 출력하면 0.0으로 수렴한 값이 출력된다. 때문에 %f 포맷이 아닌 %e 포맷을 사용해 확률값을 출력하였다.

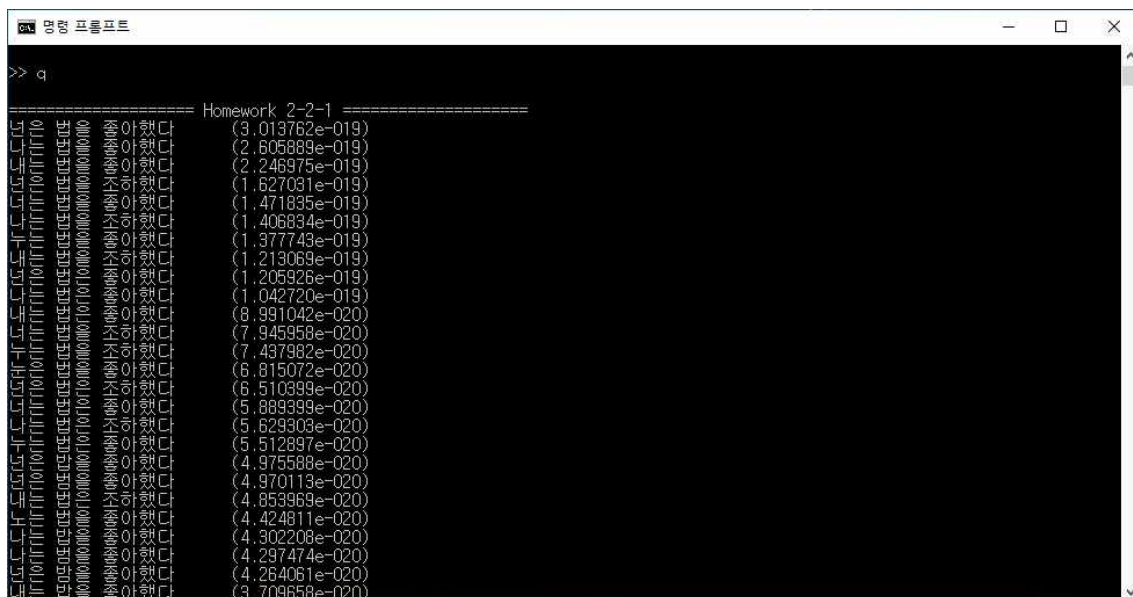
2. 실행하면 스크린샷

- Corpus: KCC940_Korean_sentences_EUCKR.txt -- preprocessing --> out.txt



```
명령 프롬프트 - calculate-sentence-percentage -2 out.txt
C:\Users\WCYS\Desktop\국민대학교\4학년 1학기\과제\빅데이터 최신기술\과제 2>calculate-sentence-percentage -2 out.txt
Total <235826> data of min. frequency 2.
===== Homework 2-1 =====
>> 나는 밥을 좋아했다
sen: 나는 밥을 좋아했다 (4.302208e-020)
>> 나는 법을 좋아했다
sen: 나는 법을 좋아했다 (2.605889e-019)
>> 너는 밥을 좋아했다
sen: 너는 밥을 좋아했다 (7.148152e-025)
>> 노는 법을 좋아했다
sen: 노는 법을 좋아했다 (2.825217e-027)
>> 사진을 찍으러 공원에 갔다
sen: 사진을 찍으러 공원에 갔다 (3.052529e-028)
>> 사진을 찍으로 공원에 갔다
sen: 사진을 찍으로 공원에 갔다 (1.921656e-024)
>>
```

키보드로부터 입력받은 문장에 대해 각 확률을 계산하였다.



```
명령 프롬프트
>> q
===== Homework 2-2-1 =====
나는 밥을 좋아했다 (3.013762e-019)
나는 법을 좋아했다 (2.605889e-019)
너는 밥을 좋아했다 (2.246975e-019)
너는 밥을 좋아했다 (1.627031e-019)
너는 법을 좋아했다 (1.471835e-019)
나는 법을 좋아했다 (1.406834e-019)
나는 법을 좋아했다 (1.377743e-019)
나는 법을 좋아했다 (1.213069e-019)
나는 법을 좋아했다 (1.205926e-019)
나는 법을 좋아했다 (1.042720e-019)
나는 법을 좋아했다 (8.991042e-020)
나는 법을 좋아했다 (7.945958e-020)
나는 법을 좋아했다 (7.437982e-020)
나는 법을 좋아했다 (6.815072e-020)
나는 법을 좋아했다 (6.510399e-020)
나는 법을 좋아했다 (5.889399e-020)
나는 법을 좋아했다 (5.629303e-020)
나는 법을 좋아했다 (5.512897e-020)
나는 법을 좋아했다 (4.975588e-020)
나는 법을 좋아했다 (4.970113e-020)
나는 법을 좋아했다 (4.853969e-020)
나는 법을 좋아했다 (4.424811e-020)
나는 법을 좋아했다 (4.302208e-020)
나는 법을 좋아했다 (4.297474e-020)
나는 법을 좋아했다 (4.264061e-020)
나는 법을 좋아했다 (3.709658e-020)
```

dic1으로부터 조합된 문장들의 생성 확률이 높은 순으로 출력되게 하였다.

결과로 “나는 법을 좋아했다”가 가장 높은 생성확률을 가지게 되었고, “누난 바블 좋아했다”가 가장 낮은 확률을 가지고 있었다.

```
Homework 2-2-2 =====
사 (1.921656e-024)
사 (3.675318e-025)
소 (1.177122e-025)
사 (9.300434e-026)
사 (8.761485e-026)
사 (4.508963e-026)
사 (1.778781e-026)
사 (1.675703e-026)
사 (1.532692e-026)
소 (5.697037e-027)
소 (5.366900e-027)
사 (4.705939e-027)
사 (4.240386e-027)
사 (2.931394e-027)
사 (2.861275e-027)
사 (2.182250e-027)
사 (2.055791e-027)
소 (1.546505e-027)
사 (9.388595e-028)
사 (9.000479e-028)
사 (8.110072e-028)
사 (7.417926e-028)
사 (7.176825e-028)
사 (6.988066e-028)
사 (6.507841e-028)
사 (5.472415e-028)
사 (4.379015e-028)
사 (3.596301e-028)
사 (3.052529e-028)
```

dic2로부터 조합된 문장들의 생성 확률이 높은 순으로 출력되게 하였다.

결과로 “사진을 찍으로 공원에 갔다”가 가장 높은 생성확률을 가지게 되었고, “사단을 짝으려 공원에 댔다”가 가장 낮은 확률을 가지고 있었다.