

# 빅데이터최신기술 과제 1 - 한글 음절빈도 계산

20171701 정지현

전체 레포지토리 링크: [https://github.com/ghyeon0/BigData\\_Homework/tree/master/HW1](https://github.com/ghyeon0/BigData_Homework/tree/master/HW1)

혹시 보고서 형식이 많이 깨진다면 [https://github.com/ghyeon0/BigData\\_Homework/blob/master/HW1/hw1.md](https://github.com/ghyeon0/BigData_Homework/blob/master/HW1/hw1.md) 를 참조 부탁드립니다.

## 1. 알고리즘

- 인코딩 구별

```
def is_utf8(text):
    lst = []
    for each in text:
        if each >= 128:
            lst.append(each)
    for i in range(0, len(lst), 3):
        if lst[i] < 0b11100000:
            return False
    return True
```

- o 아스키 코드를 제외한 모든 원소를 리스트에 넣고, 3의 간격으로 전체를 순회하면서 그 원소가 0b11100000 이상인지 검사해서 전체가 0b11100000 이상이면 utf-8로, 하나라도 틀리다면 KS 완성형으로 판정한다.

- KS 완성형인 경우 음절빈도 계산

```
def if_ks(text):
    first = ""
    second = ""
    idx = 0
    while idx < len(text):
        first = text[idx]
        idx += 1

        if first & 0x80:
            second = text[idx]
            idx += 1

        # ASCII
        if first < 128:
            freq[first] += 1

        if first >= 0xB0 and first <= 0xC8 and second >= 0xA1 and second <= 0xFE:
```

```
hfreq[first - 0xB0][second - 0xA1] += 1
```

- first에 일단 바이트 하나를 받아오고, 아스키 코드라면 freq배열에, 아스키 코드가 아니라면 바이트 하나를 더 받아와서 hfreq 배열에 카운팅한다. KS 완성형은 상위 바이트와 하위 바이트로 구성되기 때문에 위와 같은 방식으로 카운팅할 수 있다.
- UTF-8인 경우 음절빈도 계산

```
def if_utf8(text):
    first = ""
    second = ""
    third = ""
    idx = 0
    while idx < len(text):
        first = text[idx]
        idx += 1

        if first & 0x80:
            second = text[idx]
            idx += 1
            third = text[idx]
            idx += 1

            if first < 128:
                freq[first] += 1

            if first >= 0xE0 and first <= 0xEF:
                count_idx = ((first & 0x0f) << 12) | ((second & 0x3f) << 6) |
                (third & 0x3f)
                count_idx -= 0xAC00
                unifreq[count_idx] += 1
```

- KS 완성형과 방법은 비슷하지만 UTF-8은 한글을 표현하는데 3바이트가 필요하므로 아스키 코드가 아닌 경우 3개의 바이트를 받아온다.
- KS 완성형 빈도 출력

```
def ks_print():
    for i in range(25):
        for j in range(94):
            if hfreq[i][j]:
                print(bytes([i + 0xB0, j + 0xA1]).decode('cp949') + ":",
                str(hfreq[i][j]) + "회")
        print("인코딩: KS 완성형")

    for i in range(128):
        if freq[i]:
            print(chr(i), "ASCII NUM " + str(i) + ":", str(freq[i]) + "회")
```

- 전체 범위를 돌면서 하나라도 카운팅된 글자는 "안: 1회" 과 같은 방식으로 출력한다.

- 그 후에 아스키 문자에 해당되는 범위의 문자들의 빈도를 출력한다.
- UTF-8 빈도 출력

```
def utf8_print():
    for i in range(11172):
        if unifreq[i]:
            num = i + 0xAC00
            first = num >> 12
            first = 0xE << 4 | first
            second = num >> 6 & 0b111111
            second = 0b10000000 | second
            third = num & 0b111111 | 0b10000000
            print(bytes([first, second, third]).decode("utf-8") + ":",
                  str(unifreq[i]) + "회")
            print("인코딩: UTF-8")

    for i in range(128):
        if freq[i]:
            print(chr(i), "ASCII NUM " + str(i) + ":", str(freq[i]) + "회")
```

- 유니코드 한글 전체 범위를 돌면서 각 글자가 하나라도 카운팅 된 경우 utf-8 인코딩 방식과 같이 나눈 다음 "안: 1"과 같은 방식으로 출력한다.
- 그 후에 아스키 문자에 해당되는 범위의 문자들의 빈도를 출력한다.

## 2. 사용법

```
python3 hw1.py {file_name}
```

## 3. 실행결과

utf8.txt와 cp949.txt 파일은 '테스트 문장입니다.' 문장을 내용으로 하는 인코딩만 다른 파일.

```
python3 hw1.py utf8.txt
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW1 master ● python3 h
w1.py utf8.txt
니: 1회
다: 1회
문: 1회
스: 1회
입: 1회
장: 1회
테: 1회
트: 1회
인코딩: UTF-8
  ASCII NUM 32: 1회
. ASCII NUM 46: 1회
```

```
python3 hw1.py cp949.txt
```

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW1 master ● python3 h
w1.py cp949.txt
니: 1회
다: 1회
문: 1회
스: 1회
입: 1회
장: 1회
테: 1회
트: 1회
인코딩: KS 완성형
  ASCII NUM 32: 1회
. ASCII NUM 46: 1회
```

---

utf8\_ipsum.txt와 cp949\_ipsum.txt는 [한글입숨][<http://hangul.thefron.me/>] 을 통해 생성한 내용은 같고 인코딩이 다른 파일.

```
python3 hw1.py utf8_ipsum.txt
```

1. ghyeon@Ghyeons-MacBook-Pro: ~/Work/2019-1/BigData/HW1 (zsh)

```
ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW1 master • python3 h
w1.py utf8_ipsum.txt
가: 52회
간: 6회
감: 3회
갸: 1회
갓: 1회
갈: 14회
거: 5회
것: 41회
게: 18회
갸: 3회
고: 29회
곤: 2회
곳: 5회
공: 5회
과: 21회
관: 5회
광: 2회
교: 4회
구: 10회
군: 4회
곤: 4회
그: 33회
금: 9회
```

```
1. ghyeon@Ghyeons-MacBook-Pro: ~/Work/2019-1/BigData/HW1 (zsh)

품: 5회
풍: 3회
피: 21회
하: 91회
한: 24회
할: 7회
행: 1회
향: 4회
현: 7회
혜: 1회
흥: 3회
화: 2회
환: 7회
황: 10회
희: 4회
히: 3회
힘: 4회
인코딩: UTF-8

ASCII NUM 10: 8회
ASCII NUM 32: 728회
, ASCII NUM 44: 76회
. ASCII NUM 46: 86회
? ASCII NUM 63: 22회

ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW1 master
```

```
python3 hw1.py cp949_ipsum.txt
```

1. ghyeon@Ghyeons-MacBook-Pro: ~/Work/2019-1/BigData/HW1 (zsh)

ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW1 master python3 h  
w1.py cp949\_ipsum.txt

가: 52회  
간: 6회  
감: 3회  
갸: 1회  
갓: 1회  
갈: 14회  
거: 5회  
것: 41회  
게: 18회  
갸: 3회  
고: 29회  
곧: 2회  
곳: 5회  
공: 5회  
과: 21회  
관: 5회  
광: 2회  
교: 4회  
구: 10회  
군: 4회  
곧: 4회  
그: 33회  
금: 9회

```
1. ghyeon@Ghyeons-MacBook-Pro: ~/Work/2019-1/BigData/HW1 (zsh)

품: 5회
풍: 3회
피: 21회
하: 91회
한: 24회
할: 7회
행: 1회
향: 4회
현: 7회
혜: 1회
흥: 3회
화: 2회
환: 7회
황: 10회
희: 4회
히: 3회
힘: 4회
인코딩: KS 완성형

ASCII NUM 10: 8회
ASCII NUM 32: 728회
, ASCII NUM 44: 76회
. ASCII NUM 46: 86회
? ASCII NUM 63: 22회

ghyeon@Ghyeons-MacBook-Pro ~/Work/2019-1/BigData/HW1 master
```

#### 4. 전체 소스코드

```
import sys

# ASCII 카운트
freq = [0 for i in range(128)]
# KS 완성형 카운트
hfreq = [[0 for i in range(94)] for j in range(25)]
# 유니코드 한글 카운트
unifreq = [0 for i in range(11172)]

# 인코딩이 utf-8인지 검증
def is_utf8(text):
    lst = []
    for each in text:
        if each >= 128:
            lst.append(each)
    for i in range(0, len(lst), 3):
        if lst[i] < 0b11100000:
            return False
    return True
```



```

# 파일이 KS 완성형인 경우
def if_ks(text):
    first = ""
    second = ""
    idx = 0
    while idx < len(text):
        first = text[idx]
        idx += 1

        if first & 0x80:
            second = text[idx]
            idx += 1

        # ASCII
        if first < 128:
            freq[first] += 1

        if first >= 0xB0 and first <= 0xC8 and second >= 0xA1 and second <=
0xFE:
            hfreq[first - 0xB0][second - 0xA1] += 1

    ks_print()

# KS 완성형 빈도 출력
def ks_print():
    for i in range(25):
        for j in range(94):
            if hfreq[i][j]:
                print(bytes([i + 0xB0, j + 0xA1]).decode('cp949') + ":",
str(hfreq[i][j]) + "회")
    print("인코딩: KS 완성형")

    for i in range(128):
        if freq[i]:
            print(chr(i), "ASCII NUM " + str(i) + ":", str(freq[i]) + "회")

# 파일이 UTF-8 인코딩인 경우
def if_utf8(text):
    first = ""
    second = ""
    third = ""
    idx = 0
    while idx < len(text):
        first = text[idx]
        idx += 1

```

```

        if first & 0x80:
            second = text[idx]
            idx += 1
            third = text[idx]
            idx += 1

        if first < 128:
            freq[first] += 1

        if first >= 0xE0 and first <= 0xEF:
            count_idx = ((first & 0x0f) << 12) | ((second & 0x3f) << 6) |
            (third & 0x3f)
            count_idx -= 0xAC00
            unifreq[count_idx] += 1

    utf8_print()

# UTF-8 빈도 출력
def utf8_print():
    for i in range(11172):
        if unifreq[i]:
            num = i + 0xAC00
            first = num >> 12
            first = 0xE << 4 | first
            second = num >> 6 & 0b111111
            second = 0b10000000 | second
            third = num & 0b111111 | 0b10000000
            print(bytes([first, second, third]).decode("utf-8") + ":",
str(unifreq[i]) + "회")
            print("인코딩: UTF-8")

    for i in range(128):
        if freq[i]:
            print(chr(i), "ASCII NUM " + str(i) + ":", str(freq[i]) + "회")

def main(file_name):
    f = open(file_name, "rb")
    text = f.read()
    f.close()
    utf8_flag = is_utf8(text)
    if utf8_flag:
        if_utf8(text)
    else:
        if_ks(text)

```

```
if __name__ == "__main__":  
    # 실행할 때 매개변수로 파일명 지정  
    if len(sys.argv) == 2:  
        file_name = sys.argv[1]  
    # 아니면 입력 받음  
    else:  
        file_name = input("Input File Name: ")  
    main(file_name)
```