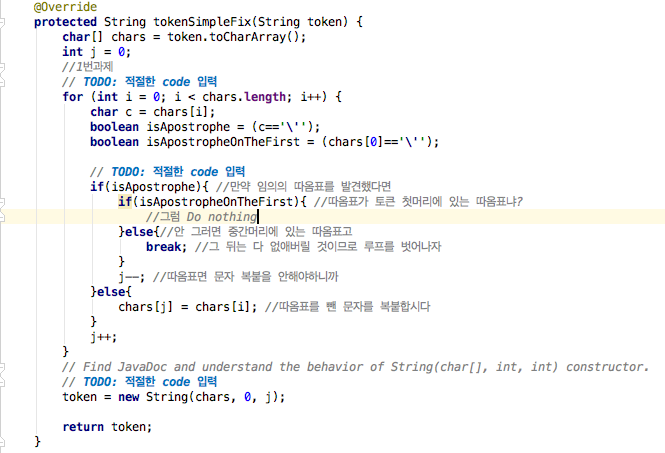
Internet Application

<HW #2>

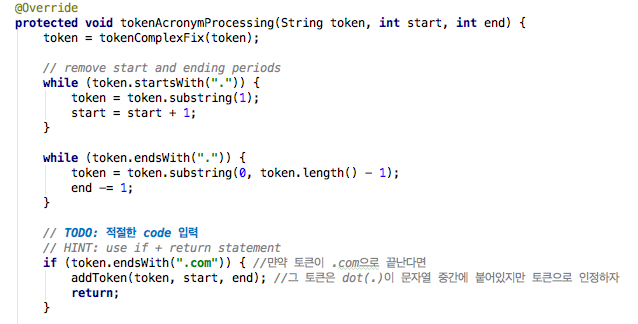
산업공학과 황영석

2010-12086

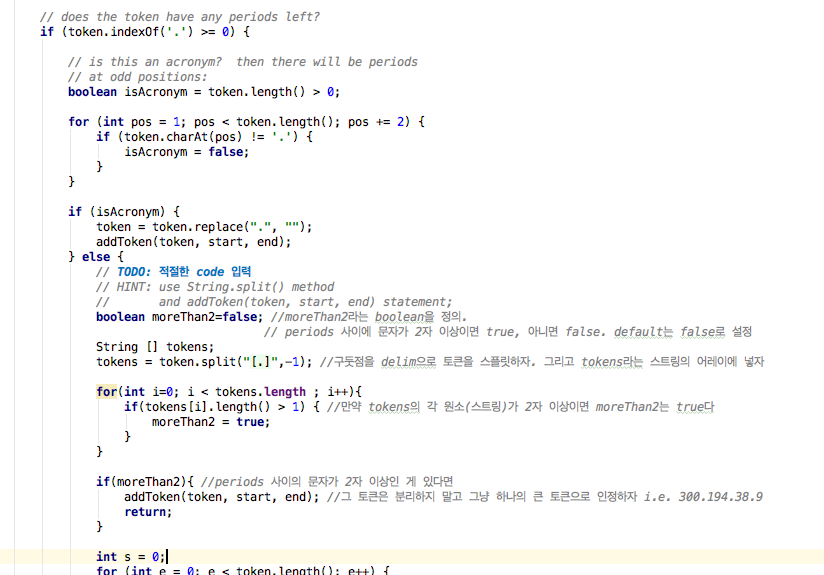
1. 번 문제 코드 및 설명(주석)
   1. **번 : 따옴표 제거**



* 1. **번 : .com 인정 (세부 토큰화 시키지 않기. 주석의 ‘토큰으로 인정하자’는 그냥 처음에 들어온 큰 토큰을 저장하자는 뜻)**



* 1. **번 : 마침표 사이 글자 2자 이상일 경우 큰 토큰 인정.**



1. 번 문제
   1. **기본 Galago**

2번 문제에서는 bible.txt를 토큰화 프로그램을 이용하여 출력한 결과를 바탕으로 ▲Zipf 분포를 그리며, ▲Zipf’s Law에서의 k값들을 구하고, ▲주어진 bible.txt가 Zipf’s Law를 따르는지 설명해야 한다. 그런데 교과서의 내용을 바탕으로 문제를 해석할 때, 분명하지 않은 점이 두 가지 있다.

첫 번째는, 교과서에 Zipf 분포는 (Rank, Freq)의 산점도로 그리는 방법과 Log-Log 스케일로 그리는 방법 두 가지가 나와 있다는 것이다. 숙제가 어떤 방법을 유도하는 건지 확실치 않아서 두 그래프 모두 그렸다.

두 번째는, 프로그램으로 출력한 데이터를 가지고서는 교과서의 Data Set 예제에 나와있는 것처럼 단어 하나하나의 Rank를 알 수 없다는 것이다. 교과서의 표 4.4를 보면 동일한 출현횟수를 보여도 Rank의 차이가 있을 수 있다. (아래 < 표 1 > 참조)

|  |  |  |
| --- | --- | --- |
| Rank | Word | Frequency |
| 1002 | summit | 5100 |
| 1003 | bring | 5099 |
| 1004 | star | 5099 |
| 1005 | iImmediate | 5099 |
| 1006 | chemical | 5099 |

< 표 > : 교과서의 표 4.4 일부

프로그램으로는 데이터가 <n번의 횟수를 보이는 단어의 수, 단어의 출현횟수 n> 형태로 나온다. 이 형태의 데이터로는 ‘n번의 횟수를 보이는 단어들’의 Rank는 동일하게 설정될 수 밖에 없다.

그리고 이 때 Rank를 설정하는 방법이 크게 2가지 있을 수 있다. 하나는 단어의 출현횟수 n을 내림차순으로 정렬한 후, 그에 따라 Rank를 부여하는 방법이다. 이렇게 rank를 설정하면, <n번의 횟수를 보이는 단어의 수, 단어의 출현횟수 n> 형태의 데이터 row가 500여 개이므로 rank의 최댓값은 500 몇 정도가 된다.

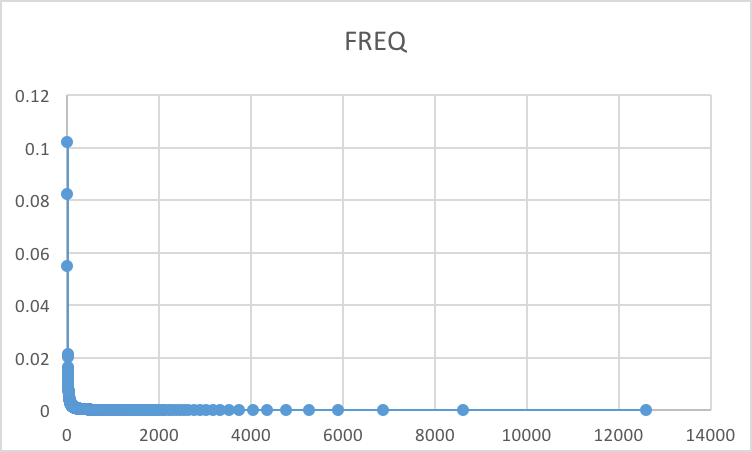
다른 방법은 n번 횟수를 보이는 단어들이 여러 개라면 그 단어의 수만큼 rank가 밀리는 것이다. 예를 들어, 위에 삽입한 표에서 Freq=5099인 단어는 총 4개이므로, 4개의 단어 bring, star, immediate, chemical은 모두 1002+4 = 1006의 rank를 동일하게 갖는다. (chemical이 Freq=5099인 단어 중 제일 마지막 단어이다.) 이 방식으로 rank를 설정하면 rank의 최대값은 단어의 총 갯수인 12600 정도가 나온다. (2.1의 경우)

이렇게 output데이터의 형태가 고정되어 있어서 Rank를 설정하는 방법이 2가지가 있을 수 있으나, 후자가 더 그럴듯한 방법이라 생각해서 본 워드파일에서는 후자의 방법을 이용하여 k값과 그래프를 나타내었다. 전자의 방법을 사용했을 때 결과도 크게는 다르지 않는데, 어쨌든 첨부한 엑셀 파일에 표시해 두었다.

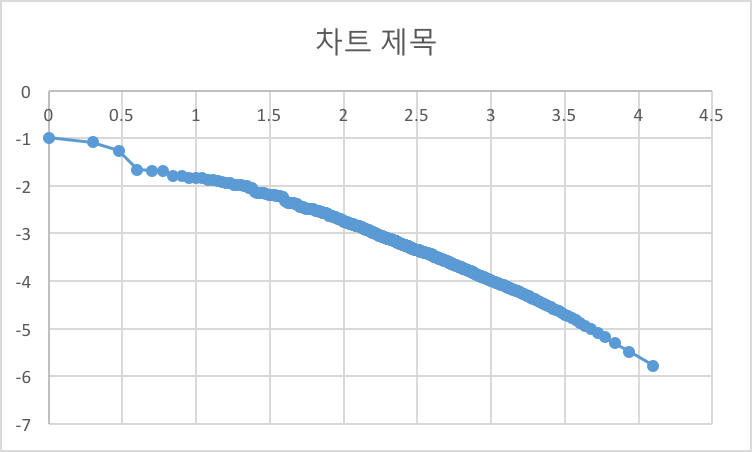
다음 페이지에는 결과 그래프를 그려놓았다.

다음은 기본 Galago를 사용하였을 때 Zipf 분포를 일반 방법과 Log-Log 스케일 방법으로 그린 그래프이다.

1. 일반 방법( 여기서 Freq는 전체 단어의 수로 나눠서 확률값으로 나타냄)



1. Log-Log 스케일

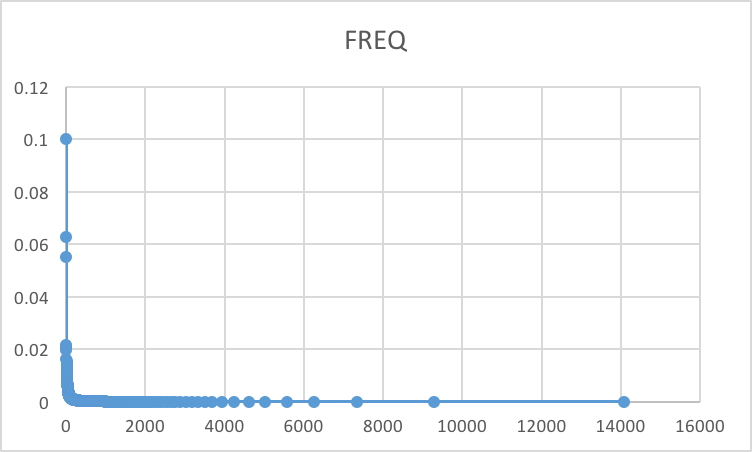


그래프를 보면 Zipf 분포를 꽤 따르는 것으로 보인다. 물론 Rank의 양 끝은 완벽히 따르지 않는 것을 확인할 수 있다.

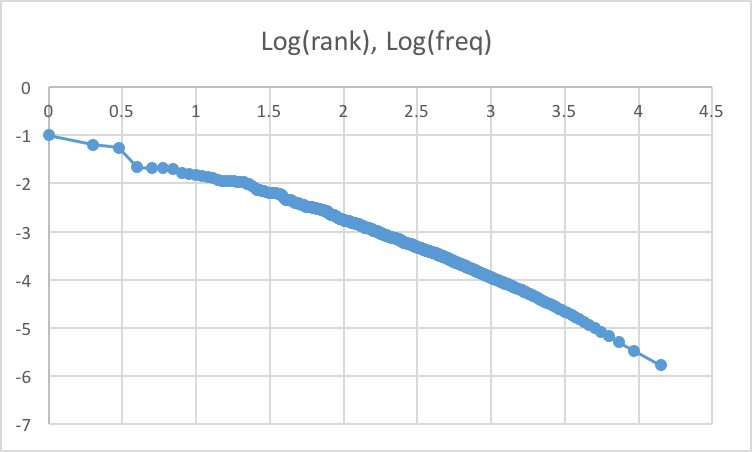
* 1. **1번 문제에서 수정한 토큰화 함수를 사용한 결과**

다음은 1번 문제에서 수정한 토큰화 함수를 사용하여 얻은 zipf 분포이다.

A. 일반 방법



B. Log-Log 스케일



2.2 역시 Zipf 분포를 꽤 따른다. 물론 Rank의 양 끝은 완벽히 따르지 않는 것 역시 확인할 수 있다.

* 1. **2.1과 2.2의 비교**

2.1과 2.2 모두 Zipf 분포를 꽤 따랐다. 각각의 자세한 k값들은 첨부한 MS 엑셀 파일에 자세히 나와 있지만, 정리하면 다음 < 표 2 >와 같다.

|  |  |  |
| --- | --- | --- |
|  | 2.1 | 2.2 |
| k 값 | 약 100,000 ± 30,000 | 약 100,000 ± 30,000 |
| 단어의 수 | 12600 | 14080 |
| 비고 | k 값은 rank 초기와 후반부에 100,000에서 많이 벗어남 | |

< 표 2 >

결과적으로 2.1과 2.2 모두 가시적인 큰 차이 없이 Zipf’s Law를 꽤 따른다고 할 수 있다.