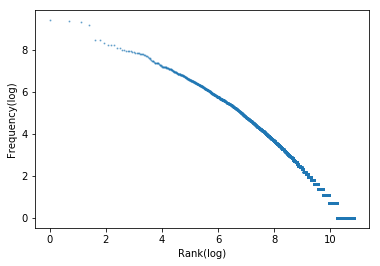
实验报告

# Zipf定律曲线生成



从图中可以看出，绘制的曲线基本符合Zipf定律，但与书上的图相比略弯曲一些，这应该是数据量小的原因。

# 基于n-gram的句子概率计算

**得到的结果如下：**

**5.227755085591562e-23 0.00032389863311207696 0.00043186484414943596 0.002152185958860412 0.0005193085522623383 0.0008137618550914991 0.002002282459238294 0.00020522502924456666**

**9.716893422249471e-15 0.00032389863311207696 0.10192837465564739 0.006198347107438017 0.012023217247097845 0.003436426116838488 0.18421052631578946 0.006238859180035651**

**4.1322035202046193e-13 0.013774104683195593**

**2.5152543166462902e-14 0.013774104683195593 0.06086956521739131**

**3.350434101246826e-23 0.00032389863311207696 0.00043186484414943596 0.002152185958860412 0.0003328214604705364 0.0008137618550914991 0.002002282459238294 0.00020522502924456666**

**3.921072366471197e-15 0.00032389863311207696 0.10192837465564739 0.006198347107438017 0.0020729684908789387 0.00804289544235925 0.18421052631578946 0.006238859180035651**

**1.667474194850066e-13 0.013774104683195593**

**1.0149842925174315e-14 0.013774104683195593 0.06086956521739131**

从中可以看出三个规律：

* Unigram的计算结果要比bigram的计算结果小很多
* 考虑句子首尾之后的影响并不是很大
* 第一句话比第二句话的频度要高。

# 实现

两个题目均使用Python实现。代码文件分别为FrequencyCalculator.py和N-GramCalculator.py