Assignment 2

CS 734: Introduction to Information Retrieval

Fall 2017

Grant Atkins

Finished on October 14, 2017

1

Question

```
4.1. Plot rank-frequency curves (using a log-log graph) for
words and bigrams in the Wikipedia collection available through
the book website (http://www.searchengines-book.com). Plot a
curve for the combination of the two. What are the best
values for the parameter c for each curve?
```

Answer

For this question I wrote two files of code, **rankFreq.py** and **rankFreq.R** in with the “small” wiki dataset provided from the textbooks website . The first python file iterates through all of the wiki html files, tokenizes them, finds token frequency, and then writes them to a CSV in descending frequency order. To retrieve the text from each of the html files I used Beautifulsoup. It should be noted when retrieving each token from the html files I did not take into account uppercase or lowercase as same terms, I treated them as different and more than probably affected the outcome of this answer. After the code tokenizing the terms it created a list of unigrams and bigrams for all the terms keeping them in separate lists to count frequencies. I used the NLTK python library to make bigram pairs. The code for this is shown below in Listing 1. The top 10, ranked by token frequency, results are shown below in Figures 1 and 2. The full CSV files can be found in my Github repository [2]. Without removing stop words its apparent that words like “the” and “of” would some of the top unigram and bigram pairs.

```
#!/usr/bin/env python3

from bs4 import BeautifulSoup
import os
import nltk
import csv


def unpackFiles():
    file_list = []
    for root, dirs, files in os.walk(os.path.dirname(”./data/en/articles”)):
        for f in files:
            if f.endswith(”.html”):
                path = os.path.join(root, f)
                file_list.append(path)

    return file_list


def tokenizeFiles(file_list):
    tokens = []
    for i, f in enumerate(file_list):
        html = open(f, 'r')
        soup = BeautifulSoup(html.read(), 'html.parser')
        text = soup.get_text()

        for word in text.split():
            if word.isalpha():
                tokens.append(word)

    return tokens


def tokenCounts(tokens):
    bigrams = list(nltk.bigrams(tokens))
    token_counts = {}
    bigram_counts = {}

    for t in tokens:
        token_counts.setdefault(t, 0)
        token_counts[t] += 1

    for t in bigrams:
        bigram_counts.setdefault(t, 0)
        bigram_counts[t] += 1

    return token_counts, bigram_counts, bigrams


def calcProbC(token_list, all_tokens):
    new_list = []
    for i, row in enumerate(token_list):
        # prob = token count / all tokens
        prob = float(row[1]) / len(all_tokens)
        # c = pos of freq in list * prob
        c = (i + 1) * prob
        new_list.append(row + [prob, c])

    return new_list


def write_csv(filename, token_type, tokens):
    with open(”./data/” + filename, 'w') as f:
        writer = csv.writer(f)
        writer.writerow([token_type, "frequency", "prob", "c"])
        writer.writerows(tokens)


def convertDimensions(token_counts):
    '''Make 2D format to write to csv'''
    d = []

    for t in token_counts:
        d.append([t, token_counts[t]])

    d = sorted(d, key=lambda x: x[1], reverse=True)
    return d


if __name__ == "__main__":
    # get all html files
    file_list = unpackFiles()
    # get list of all tokens
    tokens = tokenizeFiles(file_list)
    # count tokens. returns unigram, bigram dictionaries, bigram entire list
    tc, bc, bigrams = tokenCounts(tokens)
    # convert to sorted list based on frequeny
    t1 = convertDimensions(tc)
    t2 = convertDimensions(bc)
    # add calculations to each token(s)
    t1 = calcProbC(t1, tokens)
    t2 = calcProbC(t2, bigrams)

    write_csv("rankFreqUnigram.csv", "unigram", t1)
    write_csv("rankFreqBigram.csv", "bigram", t2)
```

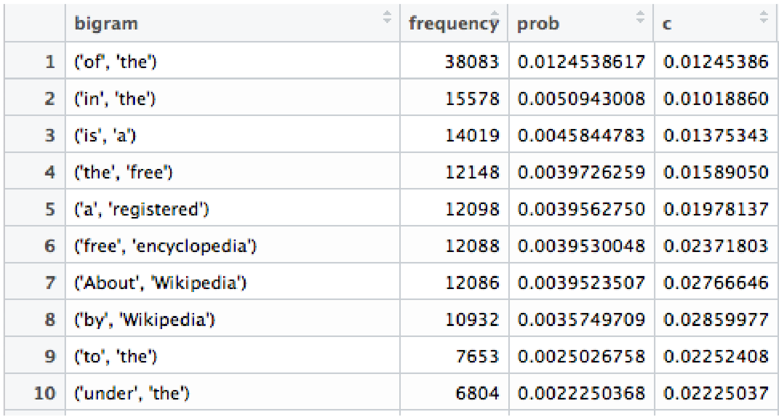
Listing 1: Python script to tokenize and find frequencies and calculate C parameters

To create the graphs I used R's ggplot2 library. The code to create these graphs is shown in Listing 2. The figures created from the afore mentioned code are shown in Figure 3, 4, and 5. For unigrams the best C parameter was 0.14, while for bigrams it was 0.1.

```
require(ggplot2)

unigramFreq <- read.csv("./data/rankFreqUnigram.csv", head = TRUE, sep = ',')
bigramFreq <- read.csv("./data/rankFreqBigram.csv", head = TRUE, sep = ',')

# unigrams
```

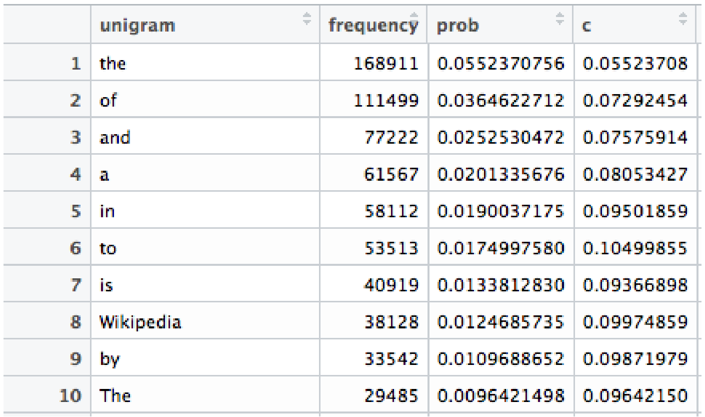
```
unigramFreq$rownum <- as.numeric(row.names(unigramFreq))

ggplot(data=unigramFreq, aes(x=rownum, y=prob)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  labs(x = "Rank", y = "Probability")

# bigrams
bigramFreq$rownum <- as.numeric(row.names(bigramFreq))

ggplot(data=bigramFreq, aes(x=rownum, y=prob)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  labs(x = "Rank", y = "Probability")

# merged graphs

ggplot(data=unigramFreq, aes(x=rownum, y=prob)) +
  geom_line(data=unigramFreq, aes(x=rownum, y=prob, color="Frequency")) +
  geom_line(data=bigramFreq, aes(x=rownum, y=prob, color="Bigram")) +
  scale_colour_manual(name='',
                      values=c('Frequency'='#5EA036', 'Bigram'='#2B56CA'),
                      guide='legend') +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = "Log-log plot of word frequency and bigrams",
       x = "Words",
       y = "Probability")
```

Listing 2: Python script to tokenize and find frequencies and calculate C parameters



	bigram	frequency	prob	c
1	('of', 'the')	38083	0.0124538617	0.01245386
2	('in', 'the')	15578	0.0050943008	0.01018860
3	('is', 'a')	14019	0.0045844783	0.01375343
4	('the', 'free')	12148	0.0039726259	0.01589050
5	('a', 'registered')	12098	0.0039562750	0.01978137
6	('free', 'encyclopedia')	12088	0.0039530048	0.02371803
7	('About', 'Wikipedia')	12086	0.0039523507	0.02766646
8	('by', 'Wikipedia')	10932	0.0035749709	0.02859977
9	('to', 'the')	7653	0.0025026758	0.02252408
10	('under', 'the')	6804	0.0022250368	0.02225037

Figure 1: Top 10 unigrams found



	unigram	frequency	prob	c
1	the	168911	0.0552370756	0.05523708
2	of	111499	0.0364622712	0.07292454
3	and	77222	0.0252530472	0.07575914
4	a	61567	0.0201335676	0.08053427
5	in	58112	0.0190037175	0.09501859
6	to	53513	0.0174997580	0.10499855
7	is	40919	0.0133812830	0.09366898
8	Wikipedia	38128	0.0124685735	0.09974859
9	by	33542	0.0109688652	0.09871979
10	The	29485	0.0096421498	0.09642150

Figure 2: Top 10 bigrams found

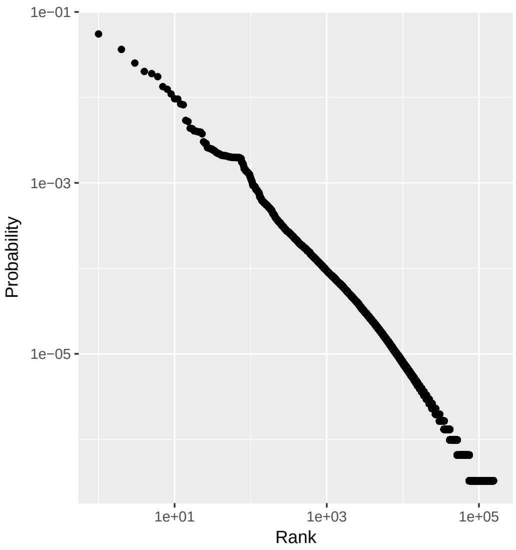


Figure 3: Log-log plot of unigram frequency and probability

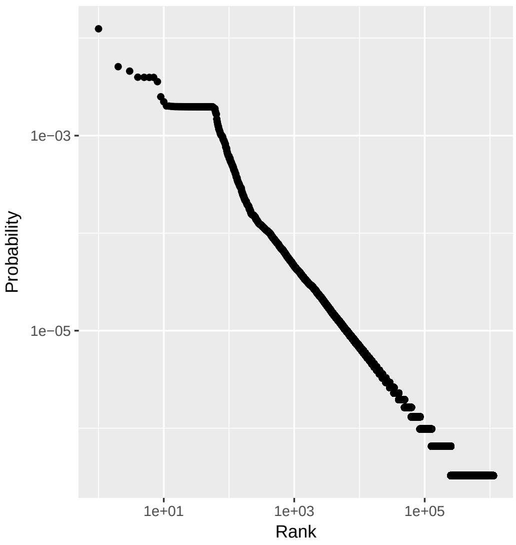


Figure 4: Log-log plot of bigram frequency and probability

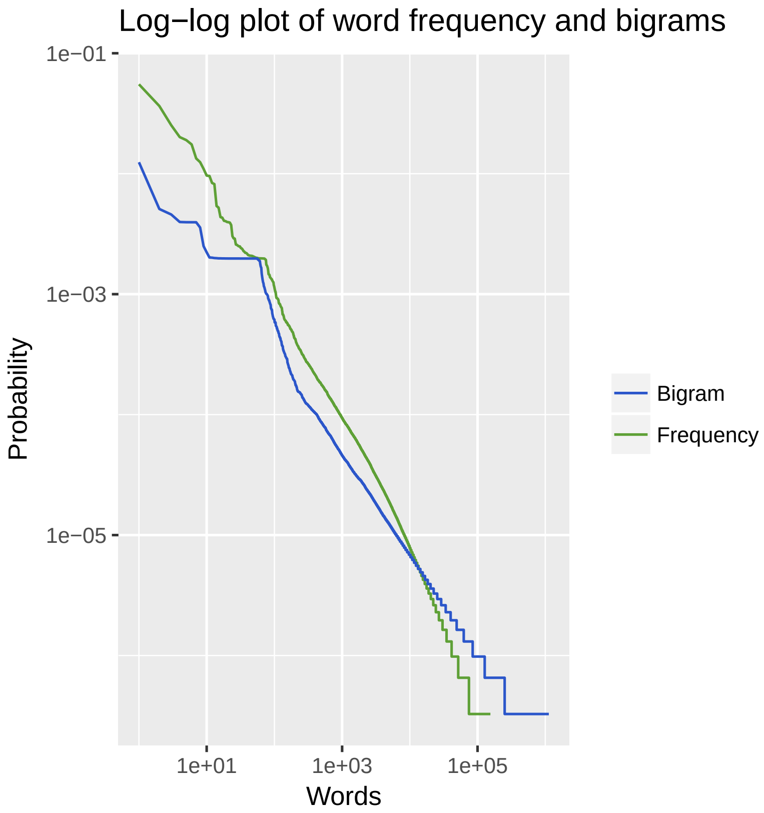


Figure 5: Log-log plot of unigram and bigram frequencies and probabilities

2

Question

```
1.4  List five web services or sites that you use that appear to use search,
not including web search engines. Describe the role of search for that
service. Also describe the search is based on a database or grep style
of matching, or if the search is using some type of ranking.
```

Answer

3

Question

3.7 Write a program that can create a valid sitemap based on the
contents of a directory on your computer's hard disk. Assume
the file are accessible from a website at the URL http://example.com.
For instance, if there is a file in your directory called homework.pdf,
this would be available at http://www.example.com/homework.pdf.
Use the real modification date on the file as the last modified time in the
sitemap, and to help estimate the change frequency.

Answer

4

Question

Suppose that, in an effort to crawl web pages faster, you set up two crawling machines with different starting seed URIs. Is this an effective strategy for distributed crawling? Why or why not.

Answer

5

Question

3.9 Write a simple single-threaded web crawler. Starting from a single input URL (perhaps a professor's web page), the crawler should download a page and then wait at least five seconds before downloading the next page. Your program should find other pages to crawl by parsing link tags found in previously crawled documents.

Answer

References

[1] Atkins, Grant. "CS532 Assignment 1 Repository" Github. N.p., 23 March 2017. Web. 23 March 2017.`https://github.com/grantat/cs532-s17/tree/master/assignments/A1/src`.

[2] Atkins, Grant. "CS734 Assignment 2 Repository" Github. N.p., 21 September 2017. Web. 21 September 2017.`https://github.com/grantat/cs834-f17/tree/master/assignments/A2`.