Introduction to Web Science

Assignment 7

Prof. Dr. Steffen Staab
staab@uni-koblenz.de

René Pickhardt
rpickhardt@uni-koblenz.de

Korok Sengupta
koroksengupta@uni-koblenz.de

Olga Zagovora
zagovora@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: December 14, 2016, 10:00 a.m.
Tutorial on: December 16, 2016, 12:00 p.m.

Please look at the lessons 1) **Similarity of Text** & 2) **Generative Models**

For all the assignment questions that require you to write code, make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.

`Delta Group Members`

Oana Dumitrasc
odumitrasc@uni-koblenz.de

Alisa Becker
alisabecker@uni-koblenz.de

Omar Aly
oaly@uni-koblenz.de

1 Modelling Text in a Vector Space and calculate similarity (10 points)

Given the following three documents:

D_1 = this is a text about web science

D_2 = web science is covering the analysis of text corpora

D_3 = scientific methods are used to analyze webpages

1.1 Get a feeling for similarity as a human

Without applying any modeling methods just focus on the semantics of each document and decide which two Documents should be most similar. Explain why you have this opinion in a short text using less than 500 characters.

Answer:

We agreed that the two most similar documents are D1 and D2 just by having a close look at the three documents and focusing on their semantics. Our theory it is based on the number of words two documents have in common. As greater the number is, the more similar the documents are.

- D1 and D2 have 4 words in common (“text”, ”is”, “web”, “science”)
- D1 and D3 have 0 in common
- D2 and D3 have 0 in common

=> The two most similar documents are D1 and D2

1.2 Model the documents as vectors and use the cosine similarity

Now recall that we used vector spaces in the lecture in order to model the documents.

1. How many base vectors would be needed to model the documents of this corpus?

   Answer: 19, for each word one (no duplicates)

2. What does each dimension of the vector space stand for?

Answer: Each dimension represents a word out of all the words in D1,D2 and D3

3. How many dimensions does the vector space have?

Answer: 19

4. Create a table to map words of the documents to the base vectors.

this	<1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
is	<0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
a	<0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
text	<0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
about	<0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
web	<0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
science	<0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
covering	<0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
the	<0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0>
analysis	<0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0>
of	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0>
corpora	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0>
scientific	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0>
methods	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0>
are	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0>
used	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0>
to	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0>
analyze	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0>
webpages	<0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1>

5. Use the notation and formulas from the lecture to represent the documents as document vectors in the word vector space. You can use the term frequency of the words as coefficients. You can / should omit the inverse document frequency.

Answer:

$$W = \{w_1, w_2, ..., w_n\}, \qquad \text{n=19} \qquad (1)$$

$$W = \{this, is, a, text, about, web, science, covering, the, analysis, of, corpora, scientific, methods, are, used, to, analyze, webpages\}, \qquad \text{n=} \qquad (2)$$

$$V =< \vec{w}_1, \vec{w}_2, ..., \vec{w}_n > \tag{3}$$

$$\vec{d}1 = \sum_{i=1}^{n} tf(w_i, D_1)\vec{w}_i \tag{4}$$

$$\vec{d}2 = \sum_{i=1}^{n} tf(w_i, D_2)\vec{w}_i \tag{5}$$

$$\vec{d}3 = \sum_{i=1}^{n} tf(w_i, D_3)\vec{w}_i \tag{6}$$

6. Calculate the cosine similarity between all three pairs of vectors.

Answer:

$$cos\,\theta = \frac{< \vec{d}_i, \vec{d}_j >}{||\vec{d}_i||_2 \cdot ||\vec{d}_j||_2} \tag{7}$$

$$\left(\vec{d}_i\right)_k = tf(w_k, D_i) \tag{8}$$

$$< \vec{d}_i, \vec{d}_j >= \sum_{k=1}^{n} \left(\vec{d}_i\right)_k \cdot \left(\vec{d}_j\right)_k = \sum_{k=1}^{n} tf(w_k, D_i) \cdot tf(w_k, D_j) \tag{9}$$

$$< \vec{d}_1, \vec{d}_2 >= \sum_{k=1}^{19} \left(\vec{d}_1\right)_k \cdot \left(\vec{d}_2\right)_k = \sum_{k=1}^{19} tf(w_k, D_1) \cdot tf(w_k, D_2) = 4 \tag{10}$$

$$< \vec{d}_1, \vec{d}_3 >= \sum_{k=1}^{19} \left(\vec{d}_1\right)_k \cdot \left(\vec{d}_3\right)_k = \sum_{k=1}^{19} tf(w_k, D_1) \cdot tf(w_k, D_3) = 0 \tag{11}$$

$$< \vec{d}_2, \vec{d}_3 >= \sum_{k=1}^{19} \left(\vec{d}_2\right)_k \cdot \left(\vec{d}_3\right)_k = \sum_{k=1}^{19} tf(w_k, D_2) \cdot tf(w_k, D_3) = 0 \tag{12}$$

$$||\vec{d}_i||_2 = \sqrt{< \vec{d}_i, \vec{d}_i >} = \sqrt{\sum_{k=1}^{n} tf(w_k, D_i)^2} \tag{13}$$

$$D1 => ||\vec{d}_1||_2 = \sqrt{< \vec{d}_1, \vec{d}_1 >} = \sqrt{\sum_{k=1}^{19} tf(w_k, D_1)^2} = \tag{14}$$

$$= \sqrt{1^2+1^2+1^2+1^2+1^2+1^2+1^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2} = \sqrt{7} \tag{15}$$

$$D2 => ||\vec{d_2}||_2 = \sqrt{<\vec{d_2},\vec{d_2}>} = \sqrt{\sum_{k=1}^{19} tf(w_k, D_2)^2} = \tag{16}$$

$$= \sqrt{0^2+1^2+0^2+1^2+0^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2} = \sqrt{9} \tag{17}$$

$$D3 => ||\vec{d_3}||_2 = \sqrt{<\vec{d_3},\vec{d_3}>} = \sqrt{\sum_{k=1}^{19} tf(w_k, D_3)^2} = \tag{18}$$

$$= \sqrt{0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2} = \sqrt{7} \tag{19}$$

$$cos\,\theta = \frac{<\vec{d_1},\vec{d_2}>}{||\vec{d_1}||_2 \cdot ||\vec{d_2}||_2} = \frac{4}{\sqrt{7}\cdot\sqrt{9}} = 0.50 \tag{20}$$

$$cos\,\theta = \frac{<\vec{d_1},\vec{d_3}>}{||\vec{d_1}||_2 \cdot ||\vec{d_3}||_2} = \frac{0}{\sqrt{7}\cdot\sqrt{7}} = 0 \tag{21}$$

$$cos\,\theta = \frac{<\vec{d_2},\vec{d_3}>}{||\vec{d_2}||_2 \cdot ||\vec{d_3}||_2} = \frac{0}{\sqrt{9}\cdot\sqrt{7}} = 0 \tag{22}$$

7. According to the cosine similarity which 2 documents are most similar according to the constructed model.

Answer: According to the constructed model D1 and D2 are the most similar, having the similarity 0.50. For D1 and D3 the similarity is zero and the same applies for D2 and D3.

1.3 Discussion

Do the results of the model match your expectations from the first subtask? If yes explain why the vector space matches the similarity given from the semantics of the documents. If no explain what the model lacks to take into consideration. Again 500 Words should be enough.

Answer:
The results do match our expectations. Which is because the Documents 1 and 2 have 4 words in common and all of these words are appearing only one time. That means that during the calculation of the document vectors, the values for 4 dimensions will be the same in comparison to none for the remaining documents. So of course when we calculate the cosine of the document vectors , d1 and d2 will be closest. And therefore result in the highest cosine similarity.

2 Building generative models and compare them to the observed data (10 points)

This week we provide you with two probability distributions for characters and spaces which can be found next to the exercise sheet on the WeST website. Also last week we provided you with a dump of Simple English Wikipedia which should be reused this week.

2.1 build a generator

Count the characters and spaces in the Simple English Wikipedia dump. Let the combined number be n. Use the sampling method from the lecture to sample n characters (which could be letters or a space) from each distribution. Store the result for the generated text for each distribution in a file.

Answer:

```
1: import collections
2: import numpy as np
3:
4: file = open('simple-20160801-1-article-per-line', 'rb')
5: simpleEnglishWiki = file.read()
6:
7: #print(type(simpleEnglishWiki))
8: countingResult = collections.Counter(str(simpleEnglishWiki))
9:
10: totNumOfChars = sum(countingResult.values())
11:
12: probsFile = open('probabilities.py.txt', 'r')
13: probs = probsFile.read()
14: probs = probs.split("\n")
15: zipf_probabilities = eval(probs[0].split("=")[1])
16: uniform_probabilities = eval(probs[2].split("=")[1])
17:
18: zipfLetters = list(zipf_probabilities.keys())
19: zipfProbs = np.fromiter(iter(zipf_probabilities.values()), dtype=float)
20: zipfCDF = np.cumsum(zipfProbs)
21:
22: uniformLetters = list(uniform_probabilities.keys())
23: uniformProbs = np.fromiter(iter(uniform_probabilities.values()), dtype=float)
24: uniformCDF = np.cumsum(uniformProbs)
25:
26:
27: zipfFile = open("zipfText.txt", "w")
28: uniformFile = open("uniformText.txt", "w")
```

```
29:
30:
31: zipfFile.write("".join(zipfLetters[np.argmax(zipfCDF > np.random.random())] for i
32: uniformFile.write("".join(uniformLetters[np.argmax(uniformCDF > np.random.random(
```

2.2 Plot the word rank frequency diagram and CDF

Count the resulting words from the provided data set and from the generated text for each of the probability distributions. Create a word rank frequency diagram which contains all 3 data sets. Also create a CDF plot that contains all three data sets.

2.3 Which generator is closer to the original data?

Let us assume you would want to creat a test corpus for some experiments. That test corpus has to have a similar word rank frequency diagram as the original data set. Which of the two generators would you use? You should perform the Kolmogorov Smirnov test as discussed in the lecture by calculating the maximum pointwise distance of the CDFs. **How do your results change when you generate the two text corpora for a second or third time? What will be the values of the Kolmogorov Smirnov test in these cases?**

Answer:

```
1: # -*- coding: utf-8 -*-
2: """
3: Created on Tue Dec 13 16:08:57 2016
4:
5: @author: Omar K. Aly
6: """
7:
8: import matplotlib.pyplot as plt
9: import collections
10: import numpy as np
11:
12: simplEngWikiFile = open("simple-20160801-1-article-per-line", "rb")
13: smplEngWiki = simplEngWikiFile.read()
14: wrdsSEW = smplEngWiki.decode("ascii", "ignore").split()
15: wrdsFreqSEW = collections.Counter(wrdsSEW)
16: wrdsCntSEW = len(wrdsSEW)
17:
18:
19: zipfTextFile = open("zipfText.txt", "r")
20: zipfText = zipfTextFile.read()
21: wrdsZipf = zipfText.split()
22: wrdsFreqZipf = collections.Counter(wrdsZipf)
```

```
23: wrdsCntZipf = len(wrdsZipf)
24:
25: uniformTextFile = open("uniformText.txt", "r")
26: uniformText = uniformTextFile.read()
27: wrdsUniform = uniformText.split()
28: wrdsFreqUniform = collections.Counter(wrdsUniform)
29: wrdsCntUniform = len(wrdsUniform)
30:
31: frequencySEW = sorted(wrdsFreqSEW.values(), reverse=True)
32: wrdsProbSEW = np.array([(x/wrdsCntSEW) for x in frequencySEW])
33: wrdsCDFSEW = np.cumsum(wrdsProbSEW)
34: rankSEW = list(range(len(frequencySEW)))
35:
36: frequencyZipf = sorted(wrdsFreqZipf.values(), reverse=True)
37: wrdsProbZipf = np.array([(x/wrdsCntZipf) for x in frequencyZipf])
38: wrdsCDFZipf = np.cumsum(wrdsProbZipf)
39: rankZipf = list(range(len(frequencyZipf)))
40:
41: frequencyUniform = sorted(wrdsFreqUniform.values(), reverse=True)
42: wrdsProbUniform = np.array([(x/wrdsCntUniform) for x in frequencyUniform])
43: wrdsCDFUniform = np.cumsum(wrdsProbUniform)
44: rankUniform = list(range(len(frequencyUniform)))
45:
46:
47: kolSmrTestSEWZipf = np.array([np.abs(wrdsCDFSEW[i]-wrdsCDFZipf[i]) for i in range
48: zipfSEWMaxPntWiseDist = np.max(kolSmrTestSEWZipf)
49:
50: kolSmrTestSEWUniform = np.array([np.abs(wrdsCDFSEW[i]- wrdsCDFUniform[i]) for i in
51: uniSEWMaxPntWiseDist = np.max(kolSmrTestSEWUniform)
52:
53: print("SEW & zipf maximum point wise distance",str(zipfSEWMaxPntWiseDist))
54: print("SEW & uniform maximum point wise distance", str(uniSEWMaxPntWiseDist))
55:
56: fig1 = plt.figure()
57: ax1 = fig1.add_subplot(111)
58:
59: fig2 = plt.figure()
60: ax2 = fig2.add_subplot(111)
61:
62: ax1.loglog(rankSEW, frequencySEW, basex=10, basey=10, linestyle = 'None',
63:            marker='.', c='b', label="Simple English Wiki")
64: ax1.loglog(rankZipf, frequencyZipf, basex=10, basey=10, linestyle='None',
65:            marker='.', c='r', label="zipf text")
66: ax1.loglog(rankUniform, frequencyUniform, basex=10, basey=10, linestyle='None',
67:            marker='.', c='g', label = "uniform text")
68:
69: ax2.plot(rankSEW, wrdsCDFSEW,
70:          c='b',label="SEW CDF")
71: ax2.plot(rankZipf, wrdsCDFZipf,
```

```
72:           c='r', label="zipf CDF")
73: ax2.plot(rankUniform, wrdsCDFUniform,
74:           c='g', label="uniform CDF")
75:
76:
77: ax1.set_xlabel("word rank")
78: ax1.set_ylabel("word frequency")
79: ax1.legend()
80:
81: ax2.set_xlabel("word rank")
82: ax2.set_ylabel("CDF")
83: ax2.set_ylim(0,1)
84: ax2.set_xscale("log")
85: handles, labels = ax2.get_legend_handles_labels()
86: ax2.legend(handles, labels, loc=2)
87:
88: plt.show()
```

2.4 Hints:

1. Build the cummulative distribution function for the text corpus and the two generated corpora
2. Calculate the maximum pointwise distance on the resulting CDFs
3. You can use `Collections.Counter`, `matplotlib` and `numpy`. You shouldn't need other libs.

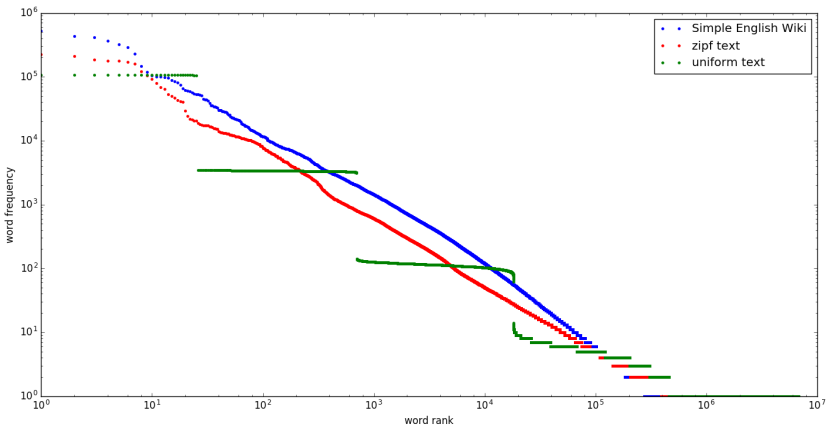

Figure 1: word rank frequency diagram 1st Run

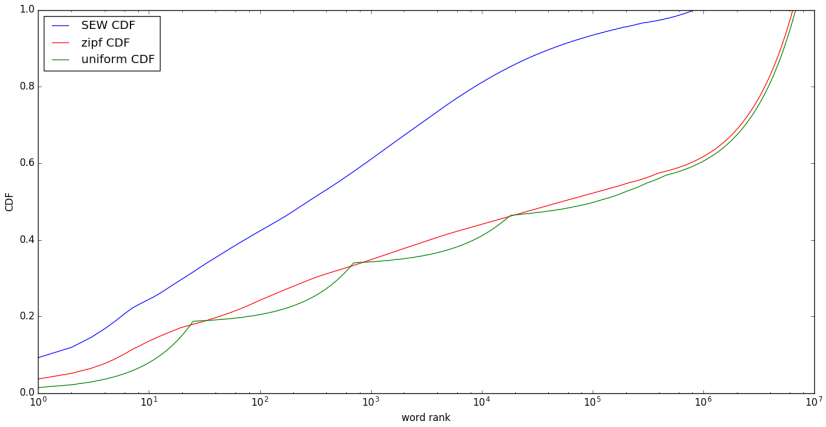

Figure 2: CDF plot 1st Run

```
SEW & zipf maximum point wise distance 0.412380401192
SEW & uniform maximum point wise distance 0.436844823566
```

Figure 3: Kolmogorov Smirnov test results 1st Run

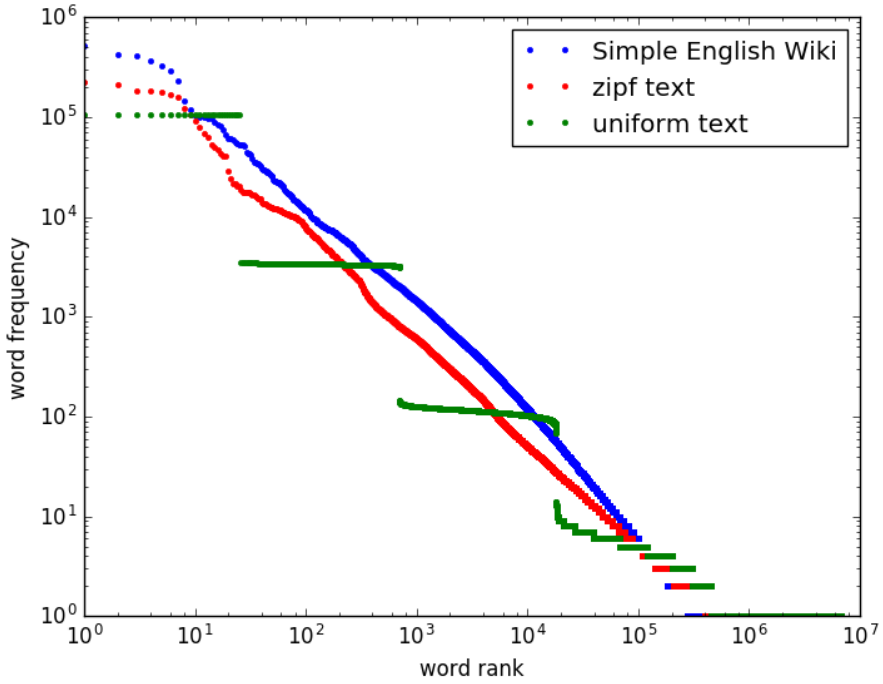

Figure 4: word rank frequency diagram 2nd Run

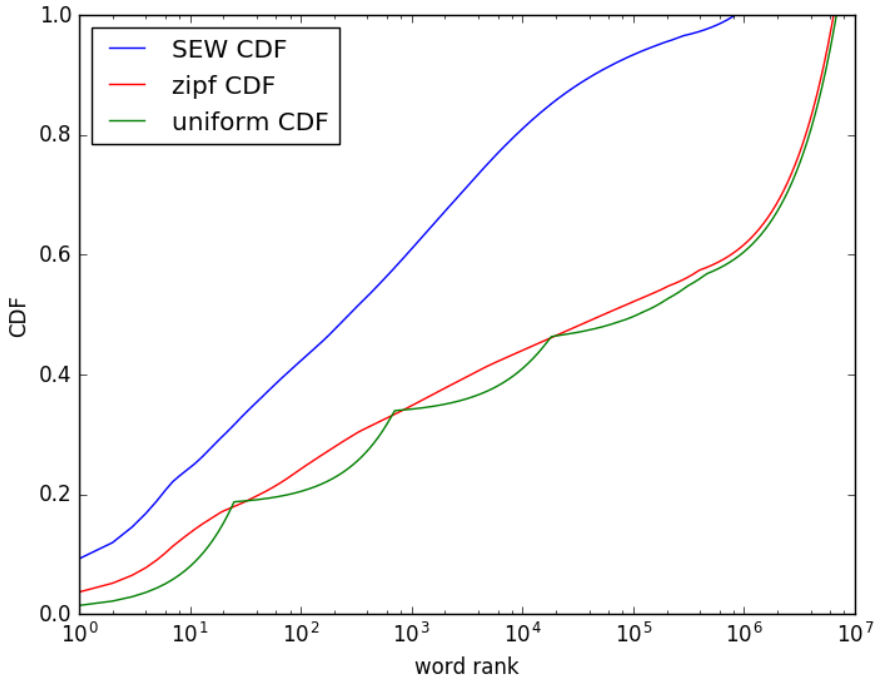

Figure 5: CDF plot 2nd Run

```
SEW & zipf maximum point wise distance 0.412463030618
SEW & uniform maximum point wise distance 0.437144183087
```

Figure 6: Kolmogorov Smirnov test results 2nd Run

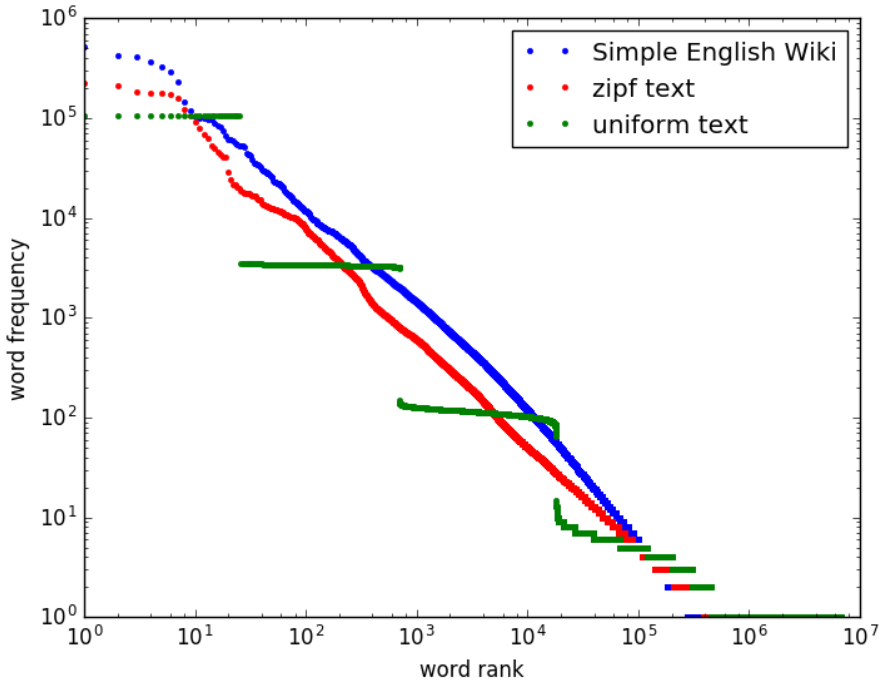

Figure 7: word rank frequency diagram 3rd Run

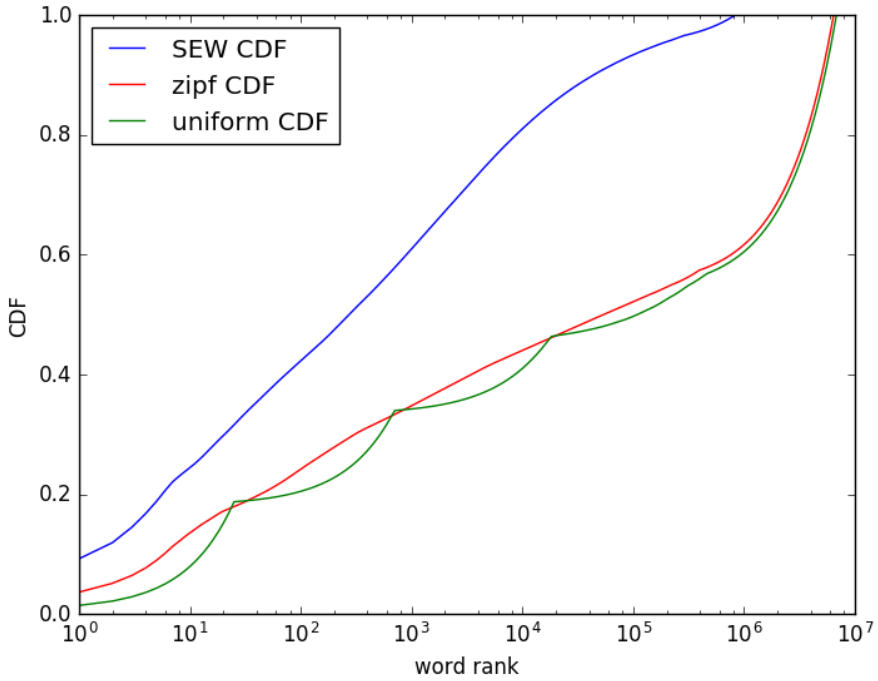

Figure 8: CDF plot 3rd Run

```
SEW & zipf maximum point wise distance 0.412514163616
SEW & uniform maximum point wise distance 0.436845263717
```

Figure 9: Kolmogorov Smirnov test results 3rd Run

3 Understanding of the cumulative distribution function (10 points)

Write a fair 6-side die rolling simulator. A fair die is one for which each face appears with equal likelihood. Roll two dice simultaneously n (=100) times and record the sum of both dice each time.

1. Plot a readable histogram with frequencies of dice sum outcomes from the simulation.
2. Calculate and plot cumulative distribution function.
3. Answer the following questions using CDF plot:

   What is the median sum of two dice sides? Mark the point on the plot.

   What is the probability of dice sum to be equal or less than 9? Mark the point on the plot.
4. Repeat the simulation a second time and compute the maximum point-wise distance of both CDFs.
5. Now repeat the simulation (2 times) with n=1000 and compute the maximum point-wise distance of both CDFs.
6. What conclusion can you draw from increasing the number of steps in the simulation?

3.1 Hints

1. You can use function from the lecture to calculate rank and normalized cumulative sum for CDF.
2. Do not forget to give proper names of CDF plot axes or maybe even change the ticks values of x-axis.

3.2 Only for nerds and board students (0 Points)

Assuming 20 groups of students. What is the likelihood that at least two groups come up with the same histograms in the case for n (=100)?
Answer:

```
1: import random
2: import numpy as np
3: import matplotlib.patches as mpatches
4: import matplotlib.pyplot as plt
5:
6:
7: def dice(n):
8:     total = 0
```

```
 9:     for i in range(0, n):
10:         total += random.randint(1, 6)
11:     return total
12:
13: def getSum():
14:     suma = list()
15:     for i in range(100):
16:         suma.append(dice(2))
17:     return suma
18:
19: def getFreqSum(suma):
20:     freqSum = {2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0, 11: 0, 12: 0
21:
22:     for i in range(0, 100):
23:         freqSum[suma[i]] += 1
24:     return freqSum
25:
26: def getCDF(freq):
27:     cumsum = np.cumsum(freq)
28:     normedcumsum = [x/float(cumsum[-1]) for x in cumsum]
29:     return normedcumsum
30:
31: suma = getSum()
32: freqSum = getFreqSum(suma)
33: # histogram with the frequencies of dice sum outcomes
34: plt.bar(list(freqSum.keys()), freqSum.values(), align='center', width=0.5)
35: plt.xticks(list(freqSum.keys()))
36: histogramLegend = mpatches.Patch(color='blue', label='Histogram')
37: plt.legend(handles=[histogramLegend], loc=5)
38: plt.xlabel("Sums")
39: plt.ylabel("Frequency")
40: plt.show()
41:
42: cdf = getCDF(list(freqSum.values()))
43:
44: cdfLegend = mpatches.Patch(color='blue', label='CDF')
45: medianLegend = mpatches.Patch(color='red', label='Median')
46: nineLegend = mpatches.Patch(color='green', label='9 mark')
47: plt.legend(handles=[cdfLegend, medianLegend, nineLegend], loc=4)
48: plt.xlabel("Sums")
49: plt.ylabel("Frequency")
50:
51: median = np.median(suma)
52: medianfreq = np.median(cdf)
53: equalOrLessThanNine = cdf[8]
54:
55: plt.axvline(median, color='b', linestyle='dashed', linewidth=2)
56: plt.axhline(medianfreq, color='r', linestyle='solid', linewidth=2)
57: plt.axhline(equalOrLessThanNine, color='g', linestyle='solid', linewidth=2)
```

```
58:
59: plt.plot(list(freqSum.keys()),cdf)
60: plt.show()
```

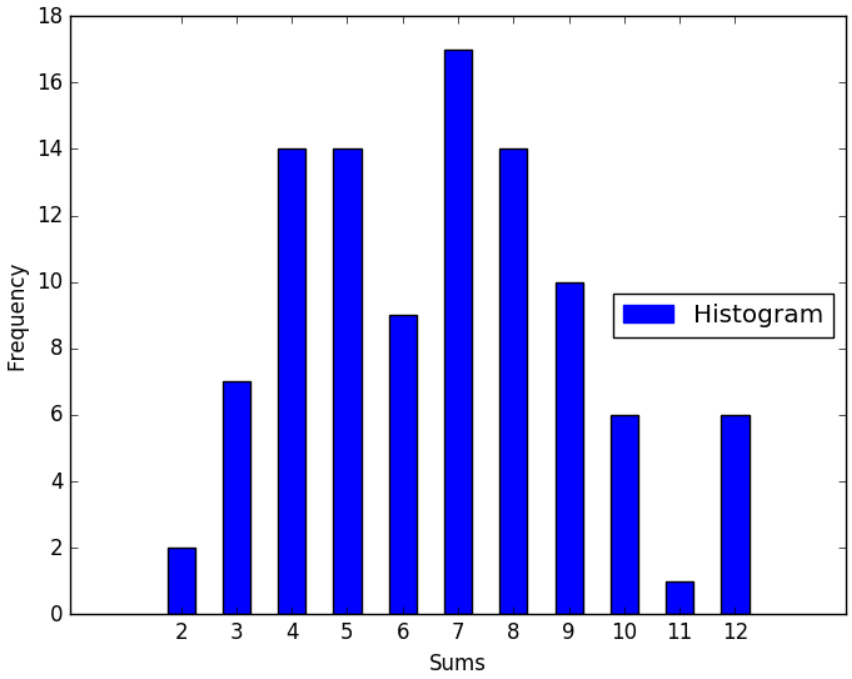

Figure 10: Histogram with frequencies of dice sum

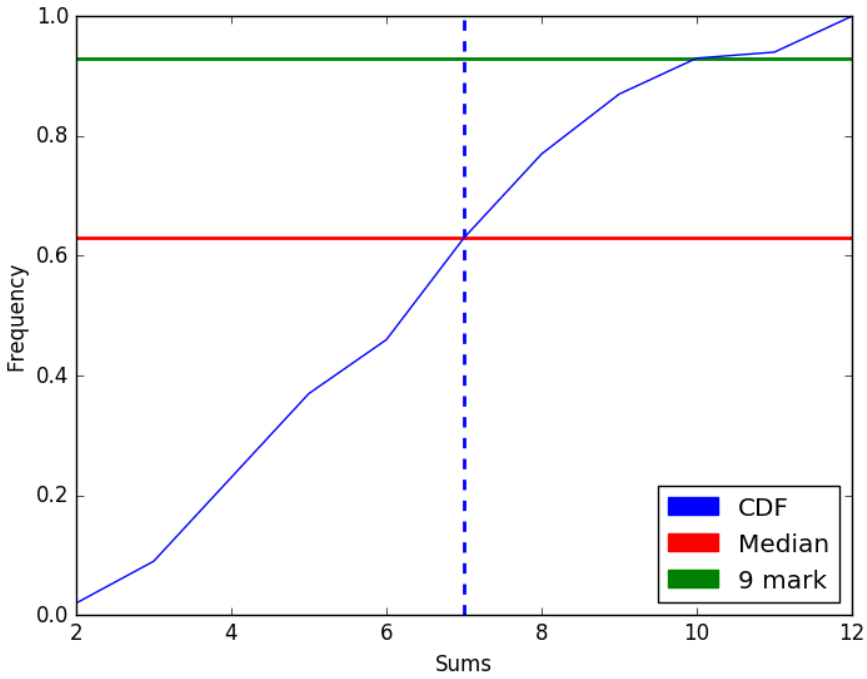

Figure 11: CDF with median sum of two dice sides and probability of dice sum = 9

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment7/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

LaTeX

Currently the code can only be build using LuaLaTeX, so make sure you have that installed. If on Overleaf, there's an error, go to settings and change the LaTeXengine to `LuaLaTeX`.