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.

Team Name: Mike
Group members: Anish Girijashivaraj, Shohel Ahamad, Slobodan Kocevski

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:
According to us in open eyes, we found most similarity between document 1 (D1) and Document 2 (D2). This is because, there are at list 4 common words which are there in both D1 and D2 (web, science, is, text). On the other hand, there is no common words between D1 and D3, or D2 and D3. Moreover, both D1 and D2 are talking about Web science and D3 is about scientific methods.

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:
   This, is,a,text, about, web, science, covering, the, analysis, of, corpora, scientific, methods,are, used, to, analyze and webpages total 19 base vectors (if we measure every single unique word is a base vector)would be required to model the documents of this corpus.
2. What does each dimension of the vector space stand for?
   Answer:
   we measure every single unique word is a base vector
3. How many dimensions does the vector space have?
   Answer:
   This vector space has 19 dimensions
4. Create a table to map words of the documents to the base vectors.
   Answer:

Table 1: Base vectors

this	1	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
a	0	0	1	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
about	0	0	0	0	1	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
science	0	0	0	0	0	0	1	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
the	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
scientific	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
methods	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
analysis	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
corpora	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
are	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
used	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
to	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
analyse	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:
   The document vector of a document D_1 is:

$$\vec{d_1} = \sum_{i=1}^{7} tf(w_i, D_1) w_i \tag{1}$$

   D1 :(1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0)
   As: this is a text about web science =
   1(1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
   + 1(0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
   + 1(0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
   +1(0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
   +1(0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0)
   +1(0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0)
   +1(0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0)
   +1(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0)
   * term frequency
   And according to this :
   D2 :(0 1 0 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0)
   D3 :(0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1)

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

$$cos(\theta) = \frac{d_1.d_2}{|d_1|.|d_2|} \tag{2}$$

   According to this :
   That the cosine similarity between d1 and d2 is = 0.50
   The cosine similarity between d1 and d3 = 0.
   The cosine similarity between d2 and d3 = 0

7. According to the cosine similarity which 2 documents are most similar according to the constructed model.
   Answer:
   According to the cosine similarity document d1 document d2 are most similar

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 result of the model matches with the expectations which we constructed earlier. Each dimensions of the vector space the words that each document consist of and it separates them into unique words refer to the unique word in the documents. the vector space model matches the similarity based on the words that each document consist of and it separates them into unique words and find the similarity between the documents so while taking the each words it more or less compared to the semantics of the document.

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:

2.1 Build a generator :

Python Code

```
import string
import random
import sys
words = lines = 0
zipf_prob = {' ': 0.17840450037213465, '1': 0.004478392057619917, '0': 0.00367182
uni_prob = {' ': 0.1875, 'a': 0.03125, 'c': 0.03125, 'b': 0.03125, 'e': 0.03125,
def makeTotal(l):
    i = 0
    n = []
    for k, v in l.iteritems():
        i += v
        n.append([i, k, v])
    return n
def getChar(l, t):
    i = 0
    n = []
    for i in range(0, len(l)):
         if(l[i][0] == t):
           return l[i][1]
def genText(l, fileName):
    ii = 0
    s = ""
    #for i in range(0, (chars+spaces)):
    for i in range(0, 100000):#for testing...
```

```
29:         v = min([row[0] for row in l], key=lambda x:abs(x-random.random()))
30:         s += getChar(l, v)
31:         ii+=1
32:     if ii >1000:
33:         sys.stdout.write('  _'+str(i)+'_    ')
34:         ii=0
35:         with open(fileName, "w") as file:
36:             file.write(s)
37: with open("articles.txt",'r', encoding="utf-8") as in_file:
38:     for line in in_file:
39:         lines += 1
40:         words += len(line.split())
41:         spaces= (words-lines)
42:
43: print ("Total number of lines are :", lines)
44: print ("Total number of words are :", words)
45: print ("Total number of spaces are :", spaces)
46: filee = open("articles.txt", "r", encoding="utf-8")
47: data=filee.read()
48: words = data.split()
49: chars = 0
50: for i in words:
51:    chars += len(i)
52: print ("Total number of characters are :", chars)
53: print ("n as mentioned, is sum of characters and spaces :", chars+spaces)
54: tot_uni_prob = makeTotal(uni_prob)
55: tot_zipf_prob = makeTotal(zipf_prob)
56: genText(tot_uni_prob, "uni.txt")
57: genText(tot_zipf_prob, "zipf.txt")
```

2.2 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.

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)?

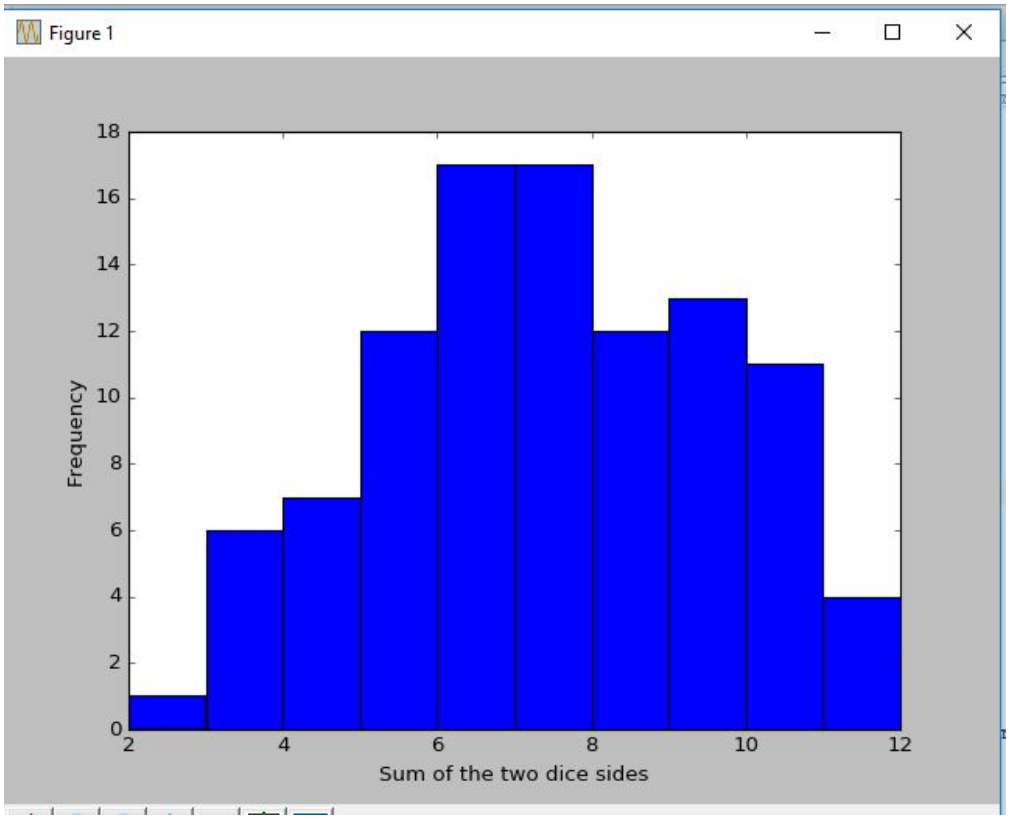
Answer
Python Code :

```
1: import random
2: import numpy as np
3: import matplotlib.pyplot as plt
4: import collections
5: def rollDiceGetSum():
```

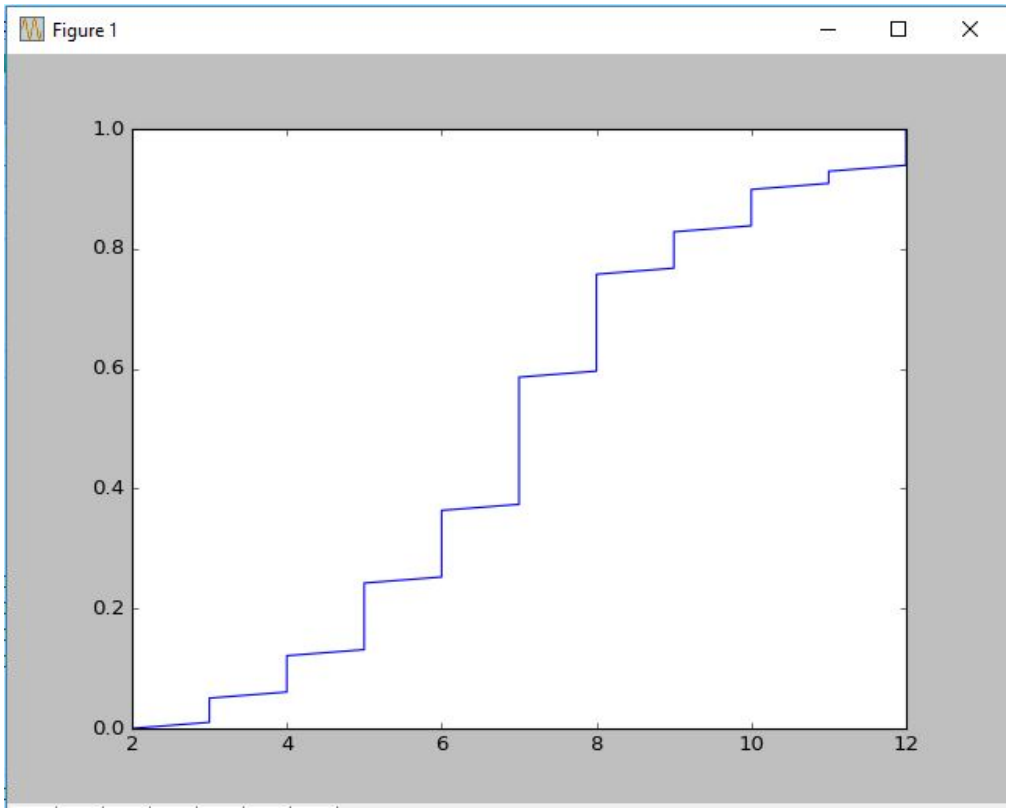
```
    return random.randint(1, 6) + random.randint(1, 6)
def calculateCDF(result):
    frequencyNum = collections.Counter(result).most_common()
    total = sum([frequency for (key, frequency) in frequencyNum])
    probabilityForEachNumDict = {}
    for (key, frequency) in frequencyNum:
        probabilityForEachNumDict[key] = frequency / \
                                         total
    arrayForCDFCalc = list(probabilityForEachNumDict.values())
    a = np.array(arrayForCDFCalc)
    cdfEvalDict = np.cumsum(a)
    return (list(probabilityForEachNumDict.keys()), cdfEvalDict)
def drawHistogram(sumResult):
    plt.hist(sumResult, bins=[2,3,4,5,6,7,8,9,10,11,12])
    plt.title('Histogram with frequencies of dice sum outcomes from the simulation
    plt.ylabel('Frequencies')
    plt.xlabel('Different Sum Results')
    plt.xticks(np.arange(2, 13, 1))
    plt.grid('on')
    plt.show()
def drawSingleCDF(sumResult, median, probLessNnine):
    (xval, cdfval) = calculateCDF(sumResult)
    plt.title("CDF PLOT for 100 times rolling two dice(sum of two dice)")
    plt.xlabel("Different Sum Results")
    plt.ylabel("Cumulative frequency in percentage")
    plt.grid('on')
    plt.xticks(np.arange(2, 13, 1))
    plt.yticks(np.arange(0, 1.1, 0.1))
    plt.plot(xval, cdfval, drawstyle='steps-post', label='CDF')
    plt.plot((2, 13), (.5, .5))
    plt.plot((median, median), (0, .5))
    plt.annotate('Median: %d' % median, (median, .5), xytext=(0.7, 0.7), arrowpro
    plt.plot((2, 13), (probLessNnine, probLessNnine))
    plt.annotate('Probability (<=9): %s' % probLessNnine, (9, probLessNnine), xyt
                 arrowprops=dict(arrowstyle='->'))
    plt.legend(loc=0)
    plt.show()
def drawDoubleCDF(result1, result2, n):
    (xval1, cdfval1) = calculateCDF(result1)
    (xval2, cdfval2) = calculateCDF(result2)
    maxDist = getMaxDistance(cdfval1, cdfval2)
    print(" The maximum pointwise distance between two CDFs for n= " + str(n)+ ",
    plt.title("CDF PLOT for " + str(n) + " times rolling two dice(sum of two dice
    plt.xlabel("Different Sum Results of rolling of two diece")
    plt.ylabel("Cumulative frequency in percentage")
    plt.xticks(np.arange(2, 13, 1))
    plt.yticks(np.arange(0, 1.1, 0.1))
    plt.plot(xval1, cdfval1, drawstyle='steps-post', label='CDF for first time')
    plt.plot(xval2, cdfval2, drawstyle='steps-post', label='CDF for second time')
```

```
55:     plt.grid('on')
56:     plt.legend(loc=0)
57:     plt.show()
58: def rollDice(n):
59:     sumResult = []
60:     for _ in range(n):
61:         result = rollDiceGetSum()
62:         sumResult.append(result)
63:     sumResult = np.sort(sumResult)
64:     return sumResult
65: def getMedian(sumResult):
66:     a = np.array(sumResult)
67:     print("Median val: ", np.median(a))
68:     return np.median(a)
69: def getProbByVal(arr, val):
70:     gen = [x for x in arr if x<= val]
71:     return (len(gen)/len(arr))
72: def getMaxDistance(result1, result2):
73:     diffResult = [abs(x - y) for x, y in zip(result1, result2)]
74:     return max(diffResult)
75: def main():
76:     sumResult = rollDice(100)
77:     drawHistogram(sumResult)
78:     median = getMedian(sumResult)
79:     val = getProbByVal(sumResult, 9)
80:     drawSingleCDF(sumResult, median, val)
81:     sumResult1 = rollDice(100)
82:     drawDoubleCDF(sumResult, sumResult1, 100)
83:     result1 = rollDice(1000)
84:     result2 = rollDice(1000)
85:     drawDoubleCDF(result1, result2, 1000)
86: if __name__ == '__main__':
87:     main()
```

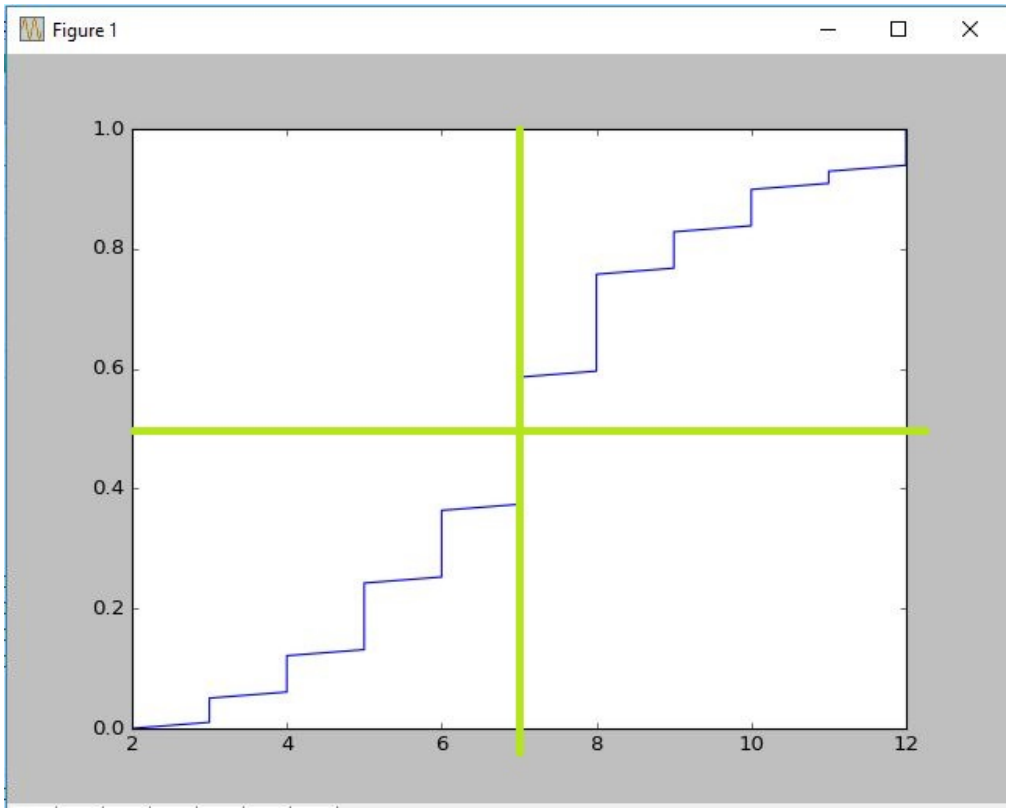
1. Histogram with frequencies of dice sum outcomes from the simulation.



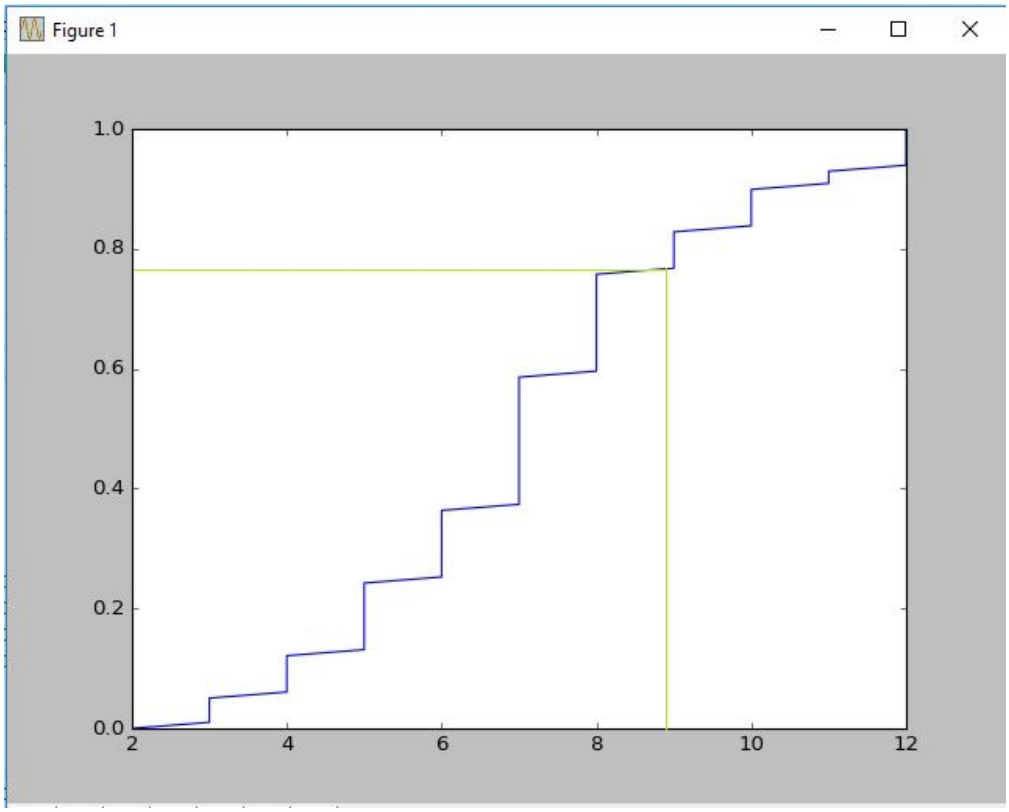
2. cumulative distribution function



3. Median sum of two dice sides is 7.



The probability of dice sum to be equal or less then 9 is 0.77



3.6

While increasing the number of steps in the simulation we got smaller value of maximum point wise distance. That means if number of experiment increase then we will get more similar CDF.

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`.