Übungsaufgaben II, SBV1

Lisa Panholzer, Lukas Fiel

November 18, 2018

1 Übungsaufgaben II

1.1 Resampling und Interpolation

a) Implementierung Resampling

b) Implementierung Bi-Lineare Interpolation

c) Implementierung Checker-Board

1.2 Klassifizierung mittels Kompression

a) Klassifizierung von Texten

Idee

Aus Texten in 8 verschiedenen Sprachen soll mittels Kompression eine Klassifizierung stattfinden. Dazu wurden folgende Datensätze als $\ddot{*}$.txt D̈atein vorbereitet:

- Abstract einer wissenschaftlichen Arbeit. Diese hatte den Vorteil dass es eine deutsche und englische Übersetzung gab. Alle weiteren Sprachen wurden aus der englischen Version mittels *google translate* generiert.
- Wörterbuch mit 10000 deutschen Wörtern. Dieser Datensatz wurde mittels *google translate* in alle anderen Sprachen übersetzt.
- Die erste Seite der Datenschutzrichtlinien von Facebook. Da die Datenschutzrichtlinien in sämtlichen Sprachen abrufbar sind, konnte für alle Sprachen ein passender Datensatz gefunden werden.
- Die erste Seite der Datenschutzrichtlinien von Google. Auch hier waren Daten in allen Sprachen verfügbar.
- Ein Witz der aus dem deutschen mittels *google translate* in alle andern Sprachen übersetzt wurde.

Da nach einer Übersetzung die Texte in verschiedenen Sprachen ungleich viele Buchstaben beinhalten ist auch die Dateigröße unterschiedlich. Dies könnte eventuell rechnerisch berücksichtigt werden. Viel einfacher aber ist es, die letzten Buchstaben jedes langen Textes zu ignorieren und so eine einheitliche Länge des Textes zu gewährleisten. Dies wurde mittels eines shell-Skripts erreicht, welches nur die ersten n Bytes eines Files speichert.

So konnte für jeden Text eine Datei erzeugt werden die in allen Sprachen den selben Speicherbedarf hat. Der Verlust der letzten Byte ist bei einer Klassifizierung unwesentlich.

```
    ; columns
#!/bin/bash


# mkdir cutTestData
mkdir cutTestData/witzData/

for filename in TestData/witzData/*.txt; do
    dd bs=1 count=8200 if="$filename" of="cut$filename"

done
```

Nach einer solchen Normierung der Texte können diese miteinander verglichen werden. Dazu wurde ein Programm in *Octave* geschrieben (siehe Listing a) , welches die Texte der einzelnen Datensätze miteinander vergleicht und in einer Matrix darstellt. Eine qualitativ hochwertige Aussage ob die Ergebisse statistische Aussagekraft haben, kann mit 5 Datensätzen nicht getroffen werden. Es ist aber sicherlich ein Trend erkennbar.

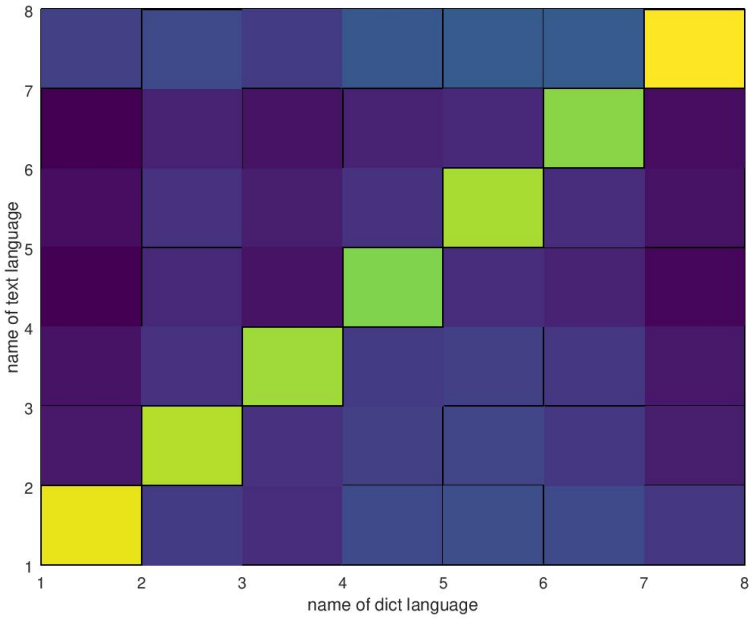


Figure 1: zipDataMatrix

Listing 1: Octave Script zur Darstellung der einzelnen Kompressionsraten.

```
    ; columns

# clear console, clear variables, close all open figures
clc
clear all
close all

# in this example we will use following languages
# languages = {"de","en","fr","es","po","un","bo","ne"};

# constants
# please note:copy-pasting windows paths: there are no backslashes in the path
folderPath = 'cutTestData/*';

# initialize result matrix
resultMatrix =zeros(8);


# loop through data folders
folders = glob(folderPath)
for x=1:numel(folders)
  [~, nameI] = fileparts (folders{x})
  for y=1:numel(folders)
    [~, nameI] = fileparts (folders{y})

    # just calculate matrix for different data sets
    if (!strcmp(folders{x},folders{y}))

      folderPath1 = [folders{x} , '/*'];
      folderPath2 = [folders{y} , '/*'];
      filesOfFolder1 = glob(folderPath1);
      filesOfFolder2 = glob(folderPath2);

      # for every element in data folder
      for i=1:numel(filesOfFolder1)
        [~, nameI] = fileparts (filesOfFolder1{i});
        #get file size
        [info, err, msg] = stat (filesOfFolder1{i});
        file1Size = info.size;

        #CREATE zip of file
        tmpFolderPath = nameI;
        mkdir(tmpFolderPath);
        copyfile(filesOfFolder1{i},tmpFolderPath);
        firstZipName = [nameI, '.zip']
        zip(firstZipName,[tmpFolderPath,'/*']);
        #get zip size
        [info, err, msg] = stat (firstZipName);
        file1file2ZipSize = info.size;

        # calculate compression rate
        file1CompressionRate = file1file2ZipSize / file1Size;

        #delete zip and folder as we just need the size for calculation
        delete([tmpFolderPath,'/',nameI,'.txt']);
        rmdir(tmpFolderPath);
        delete(firstZipName);

        for j=1:numel(filesOfFolder2)
          [~, nameJ] = fileparts (filesOfFolder2{j});
          zipName = [nameI,nameJ,'.zip'];
          #create tmp folder in testData folder
          tmpFolderPath = [nameI,nameJ];
          mkdir(tmpFolderPath);

          #copy filesOfFolder1 to folder
          ['copy filesOfFolder1 ', nameI, ' , ' nameJ , ' to ' , tmpFolderPath];
          copyfile(filesOfFolder1{i},tmpFolderPath);
          copyfile(filesOfFolder2{j},tmpFolderPath);

          # get folder size (add jokeFile size to filesOfFolder1ize)
          [info, err, msg] = stat (filesOfFolder2{i});
          file2Size = file1Size + info.size;

          #zip it and get size of zip
          zip(zipName,[tmpFolderPath,'/*']);
```

```
          [info, err, msg] = stat (zipName);
          file2ZipSize = info.size;

          #calculate compression rate
          compressionRate = file2ZipSize / file2Size ;

          #calculate exprected rate
          expectedfile2ZipSize = file2Size * file1CompressionRate;

          %Matrix(i,j) = (expectedfile2ZipSize - file2ZipSize) /  file2ZipSize;
          kompressionDict = (file1Size -file1file2ZipSize) / file1Size;
          kompressionBoth = (file2Size - file2ZipSize) / file2Size;
          kompressionsDelta = abs(kompressionBoth - kompressionDict);
          Matrix(i,j) = kompressionsDelta;


          #cleanup - remove tmp folder
          ['remove ' tmpFolderPath];
          delete([tmpFolderPath,'/',nameI,'.txt']);
          delete([tmpFolderPath,'/',nameJ,'.txt']);
          rmdir(tmpFolderPath);
          delete(zipName);
        endfor
      endfor

      resultMatrix = resultMatrix .+ Matrix;

    else # dont calculate anything if the folders are the same
      ["skip " folders{x}]
    endif


  endfor
endfor

    figure()
    surf(resultMatrix)
    view(2)
    xlabel("name of dict language");
    ylabel("name of text language");
    zlabel(" diff of compression rates");
    'SUCCESS'
```

b) OPTIONAL – nur für Interessierte/Experten

1.3 Kompression und Code-Transformation

a) Lempel-Ziv Kompression einer Sequenz

Figure a) zeigt die händische Berechnung der Lemper Ziv Kompression. Beim Übertragen ins Protokoll wurde allerdings ein Fehler entdeckt, der in Tabelle 1 korrigiert wurde.

Aufgabe 2.3.a Lempel Ziv Kompression

Zeichenkette: a b a b a b b b a a a a b a b a b c c d d d a

aktuelles Zeichen	nächstes Zeichen	Ausgabe? im Wörterbuch?		Ins Wörterbuch?	
a	b	N	a	ab	256
b	a	N	b	ba	257
a	b	Y	256	aba	258
~~a~~	b	Y	256	abb	259
b	b	N	b	bb	260
b	a	Y	257	baa	261
a	a	N	a	aa	262
a	a	Y	262	aab	263
a	a	Y	257	bab	264
b	a	Y	257 Y 264	babc	265
c	c	N	c	cc	266
c	d	N	c	cd	267
d	d	N	d	dd	268
d		Y	268	dda	269

23 Zeichen im Ursprungstext
14 Zeichen im Resultat

Zeichenkette:

97, 98, 256, 256, 98, 257, 97, 262, 257, 257, 264, 99, 99,
↳ 100, 268

Kompressionsrate: $\frac{23}{14} = 1,64$

aktuelles Zeichen	nächstes Zeichen	Ausgabe (im Wörterbuch?)	ins Wörterbuch!	Speicher
a	b	N → a	ab	256
b	a	N → b	ba	257
a	b	Y → 256	aba	258
a	b	Y → 256	abb	259
b	b	N → b	bb	260
b	a	Y → 257	baa	261
a	a	N → a	aa	262
a	a	Y → 256	aab	263
b	a	Y → 257	bab	264
b	a	Y (257), Y → 264	babc	265
c	c	N → c	cc	266
c	d	N → c	cd	267
d	d	N → d	dd	268
d	d	Y → 268	dda	269
a		N → a		

Table 1: Level Ziv Kompression

In der korrigierten Version ergibt die resultierende Zeichenkette:

97	98	256	256	98	257	97	262	257	264	99	99	100	268	269

$$[H]KompressionsrateC = \frac{23}{15} = 1.5334 \tag{1}$$

b) Huffmann Coding

Die nächste Seite zeigt die händische Berechnung des Huffmann Baums zum gegebenen Beispiel. Die mittlere Codewortlänge lieght dabei bei $1.869bit$.

Weiters kann man auf der darauffolgenden Seite die manuelle Berechnung von 6 Testfällen finden. Es kann gesagt werden, dass die Huffmann Kompression sehr gut geeignet wäre um Daten zu komprimieren, die sehr oft gleich sind. Dabei spielt homogenität keine Rolle, sondern rein die Häufigkeit des Auftretens der Sonderfälle. An diser Stelle sei erwähbt, dass die Information

über solche Sonderfälle nicht verlohren geht, sondern einfach mehr Speicher benötigt (verlusstfreies Komprimieren)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a	b	a	b	a	b	b	b	a	a	a	a	b	a	b	a	b	c	c	d	d	d	a

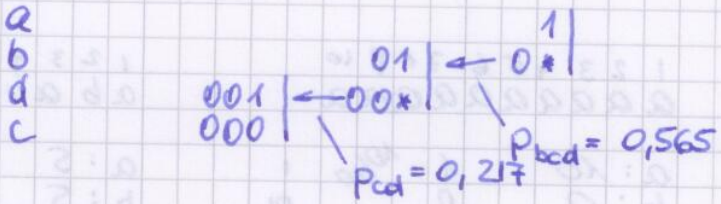
codierung $a = 00,\ b = 01,\ c = 10,\ d = 11$

23 Zeichen zu 2 bit ergeben eine Länge von 46 bit

Häufigkeiten

a: 10	$10/23 = 0{,}435$	} sort {	a	1	1
b: 8	$8/23 = 0{,}348$		b	01	2
c: 2	$2/23 = 0{,}087$		c	000	4
d: 3	$3/23 = 0{,}130$		d	001	3

Baum



mittlere Codewortlänge

$$L = 1 \cdot 10 + 2 \cdot 0{,}348 + 3 \cdot 0{,}087 + 3 \cdot 0{,}130 = 1{,}782 \text{ bit}$$

Kompressionsrate $\frac{2 \text{ bit}}{1{,}782 \text{ bit}} = 1{,}122$

Binärlayout: $10 \cdot 1 + 8 \cdot 2 + 2 \cdot 3 + 3 \cdot 3 = 41$ bit

Zeichen: {a, b, c, d} → 2bit Darstellung {00, 01, 10, 11}

1 2 3 4 5 6 7 8 9 10
a b c d a b c d a b

a: 3	$3/10 = 0,3$	1
b: 3	$3/10 = 0,3$	01
c: 2	$2/10 = 0,2$	001
d: 2	$2/10 = 0,2$	000

$L = 1 \cdot 0,3 + 2 \cdot 0,3 + 3 \cdot 0,2 \cdot 2$
$= 2,1$

Kompressionsrate: 0,95
→ keine Kompression

1 2 3 4 5 6 7 8 9 10
a a a a a a a b c d

a: 7	$7/10 = 0,7$	1
b: 1	$1/10 = 0,1$	01
c: 1	$1/10 = 0,1$	001
d: 1	$1/10 = 0,1$	000

$L = 1 \cdot 0,7 + 2 \cdot 0,1 + 3 \cdot 0,1 \cdot 2$
$= 1,5$

Kompressionsrate: ~~1,5~~ 1,3

1 2 3 4 5 6 7 8 9 10
a a a a a a a a a a

a: 10	$1 = 10/10$	1
b: 0	0	01
c: 0	0	001
d: 0	0	000

$L = 1 \cdot 1 + 2 \cdot 0 + 3 \cdot 0 \cdot 2$
$= 1$

Kompressionsrate: 2

1 2 3 4 5 6 7 8 9 10
a b a b a b a b a b

a: 5	$5/10 = 0,5$	1
b: 5	$5/10 = 0,5$	01
c: 0	0	001
d: 0	0	000

$L = 1 \cdot 0,5 + 2 \cdot 0,5 + 3 \cdot 0$
$= 1,5$

Kompressionsrate: 1,3

1 2 3 4 5 6 7 8 9 10
b b b b b c b b b b

a: 0	0	000
b: 9	$9/10 = 0,9$	1
c: 1	$1/10 = 0,1$	01
d: 0	0	001

$L = 1 \cdot 0,9 + 2 \cdot 0,1 + 3 \cdot 0 \cdot 2$
$= 1,1$

Kompressionsrate: 1,82

1 2 3 4 5 6 7 8 9 10
c d c d c d c d c d

a: 0	0	000
b: 0	0	001
c: 5	$5/10 = 0,5$	01
d: 5	$5/10 = 0,5$	1

$L = 1 \cdot 0,5 + 2 \cdot 0,5 + 3 \cdot 0 = 1,5$

Kompressionsrate: 1,3

c) Kompression einer Sequenz

d) Entropieberechnung