Welcome to the Statistical Methods of Language Technologyb SoSe21 course

Dr. Seid Muhie Yimam

- Email: yimam@informatik.uni-hamburg.de (mailto:yimam@informatik.uni-hamburg.de)
- Office: Informatikum, F-415

Dr. Özge Alaçam

- Email: alacam@informatik.uni-hamburg.de (mailto:alacam@informatik.uni-hamburg.de)
- Office: Informatikum, F-435

Topic of this week;

In this first practice class, we are going to focus on two main topics, which will be useful to complete the assignment;

- Latent Semantic Analysis (LSA)
- Latent Dirichlet Allocation (LDA)

Deadline: 16/21 June

Preperation: R installation

The exercises can be done with R, but you can also use any programming language. A small introduction into R can be found in moodle.

- Download and install R (http://www.r-project.org/ (http://www.r-project.org/)) current version 4.1.0 or you can also use R with Anaconda installation here https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/ (https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/)
- For Linux, you might need to add a repository, e.g. https://cran.uni-muenster.de/ (https://cran.uni-muenster.de/)
- You can also install RStudio Desktop (https://rstudio.com/products/rstudio/download/#download (https://rstudio.com/products/rstudio/download/#download))
- Or having an installation in your own computer is recommended, but if it does not work, you can access to our R server https://ltdemos.informatik.uni-hamburg.de/rserver/ (https://ltdemos.informatik.uni-hamburg.de/rserver/) with your LDAP credentials.

After installation, you can run R by executing:

`$ R` in the command line terminal.

We will also need a package called tm (text mining). This can be installed within R by typing: `> install.packages("tm",dep=T)`

If you want to install it systemwide open R as Administrator. R can be closed by typing `q()` or pressing **Ctrl+D**. If you want to use functions of the *tm package* you have to load it. This can be done by typing:

```
> library(tm)
```

In class Exercises

Problem 8.1 Latent Semantic Analysis (LSA)

a) Download and extract the additional data from moodle. The archive contains the training documents for this exercise. All files start with *c* (for corpus), the second letter indicates the *topic* of the document.

b) Read the training files:

- load the text mining package: `library(tm)`
- load the names of the training documents: `files = DirSource(".",pattern="^c.*")`
- create a corpus and read the files: `training=Corpus(files)`
- to get an impression of the training object type: `training`
- to get a deeper look into the documents type: `inspect(training)`
- to convert the documents to a document term matrix use: `dtm = DocumentTermMatrix(training,control=list(tolower=F))`

c) List the number of times and the percentage of following terms within the corpus and the percentage of occurrence within a document : “bank”, “money”, “stream” , “river”

Hint: use `as.matrix(dtm)` to convert the document term matrix to a “real matrix”

```
m = as.matrix(dtm)
sum(m[,"bank"])
 sum(m[,"money"])
sum(m[,"stream"])
sum(m[,"river"])
sum(m[,"bank"])/sum(m)
sum(m[,"money"])/sum(m)
sum(m[,"stream"])/sum(m)
sum(m[,"river"])/sum(m)  \
```

- Count how often the number is greater than 0 and divide it by the number of columns/documents \

```
dim(m[m[,"bank"]>0,])[1] / length(m[,1])
dim(m[m[,"money"]>0,])[1] / length(m[,1])
dim(m[m[,"river"]>0,])[1] / length(m[,1])
dim(m[m[,"stream"]>0,])[1] / length(m[,1])  \
```

d) Plot the first two dimensions of the document matrix using singular value decomposition

Hints:

- The R command for SVD is `svd(Matrix)` (lookup the command `?svd`). `svd(Matrix)` returns an object with values `u,s,v` which can be accessed using:

```
s = svd(Matrix)
s$u;s$v;s$d
```

- If you want to transpose a matrix use `t(matrix)`
- Matrix multiplication is done by `V %*% U`
- You can use the provided plot function `plotLatentVariables}`

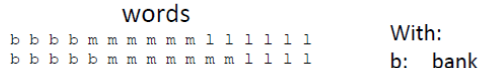
Plot method

```
plotLatentVariables<-function(val,files,showFilenames=F){
    f = substr(basename(files$filelist),0,2)
    t = unique(f)
    s = match(f,t)
    plot(val,pch=s,col=s)
    legend("topleft", legend=t, pch=1:length(t),col=1:length
(t))
    if(showFilenames){
        text(val, labels=basename(files$filelist))
        }
    }
```

apply svd on the document term matrix

```
r=svd(dtm)
 u = r$u
 s = diag(r$d)
 v = r$v
 plotLatentVariables (u, files )
 plotLatentVariables (u, files, showFilenames = TRUE)
```

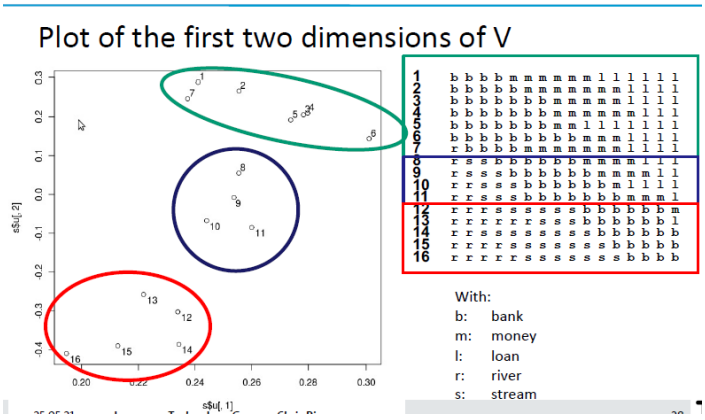
Hints from lecture Slides:



- Perceive Corpus as VSM

ID	b	r	s	m	l
1	4	0	0	6	6
2	5	0	0	7	4
3	7	0	0	5	4
4	7	0	0	6	3
5	7	0	0	2	7
6	9	0	0	3	4
7	4	1	0	7	4
8	6	1	2	4	3
9	6	1	3	4	2
10	6	2	3	1	4
11	7	2	3	3	1
12	6	3	6	1	0
13	6	6	3	0	1
14	6	2	8	0	0
15	5	4	7	0	0
16	4	5	7	0	0

With:
b: bank
m: money
l: loan
r: river
s: stream



R cheatsheet

R plot pch symbols : The different point shapes available in

Different plotting symbols are available in R. The graphical argument used to specify point shapes is pch.

```
pch = 0,square
pch = 1,circle
pch = 2,triangle point up
pch = 3,plus
pch = 4,cross
pch = 5,diamond
pch = 6,triangle point down
pch = 7,square cross
pch = 8,star
pch = 9,diamond plus
pch = 10,circle plus
pch = 11,triangles up and down
pch = 12,square plus
pch = 13,circle cross
pch = 14,square and triangle down
pch = 15, filled square
pch = 16, filled circle
pch = 17, filled triangle point-up
pch = 18, filled diamond
pch = 19, solid circle
pch = 20,bullet (smaller circle)
pch = 21, filled circle blue
pch = 22, filled square blue
pch = 23, filled diamond blue
pch = 24, filled triangle point-up blue
pch = 25, filled triangle point down blue
```

Problem 8.2 Latent Dirichlet Allocation

a) Download, unpack and compile GibbsLDA++ (http://gibbslda.sourceforge.net/ (http://gibbslda.sourceforge.net/)) into the corpus directory. **Note:** on some systems, you need to add `#include <cstdlib>` to util.cpp, and `#include <stdio.h>` to lda.cpp for compilation) \

```
$ tar -xzvf GibbsLDA++-0.2.tar.gz

$ cd GibbsLDA++-0.2

$ make clean all
```

If you encounter problems during compilation, you will need to add dependencies to files in `./src/`

- **utils.cpp** `#include <cstdlib>`
- **lda.cpp** `#include <stdio.h>`

b) Convert the training documents from ??? into the format for \mbox{GibbsLDA} with the following specification:

GibbsLDA File Format Specification: \

Specification:	Example:
N	`3`
$w_{11} \dots w_{1m}$	`I have no money on my bank account`
$\dots$	`More than one ship got stuck on that bank in the river`
$w_{N1} \dots w_{Nk}$	`There is money in the river`

- run the following command inside the "PS-08-data" folder \

```
f=lda-input.txt; ls c* | wc -l | tr -d " " > $f; cat c* >> $f
```

c) Train GibbsLDA with 3 topics (as we have 3 different kinds of documents). Inspect all output files.

```
GibbsLDA++-0.2/src/lda -est -beta 0.01 -ntopics 3 -niters 500
-twords 30 -dfile lda-input.txt
```

Output files:
model-final.twords: contains most likely words per topic
model-final.tassign: contains the topic assignment of each word in the training data
model-final.theta: topic-document distribution
model-final.phi: word-topic distribution
wordmap.txt: vocabulary with index of each word

d) Compare the topic distribution of each document with every other document and plot the result.

Use the following distance function between two documents x and y (represented as vectors of topic probabilities), where K is the number of topics:

$$d(x, y) = \sqrt{\sum_{k=1}^{K} (x_k - y_k)^2}$$

Briefly describe what you see. \

```
library(tm)
 t = read.table("model-final.theta")
 f = DirSource(".",pattern="^c.*")
 fp = substr(basename(f$filelist),0,2)
 color = rainbow(length(unique(fp)))
 ndoc = dim(t)[1]
 #initialize plot
 plot(c(),xlim=c(1,ndoc),ylim=c(0,1),xaxt="n", xlab="Documen
t", ylab="Similarity")
 sim=matrix(0,ndoc,ndoc)
 for(i in 1:ndoc){
   fpi = which(unique(fp)==fp[i])
   v1=t[i,]
  for(j in 1:ndoc){
    v2=t[j,]
    sim[i,j]=1-sqrt(sum((v1-v2)^2))
```

Hints from lecture Slides:

- Example: We want to train 2 topics with α=1 and β=0.1

Doc 1	a	a	b	b	a	a
Doc 2	a	b	b	c	c	a
Doc 3	d	d	d	c	c	
Doc 4	d	c				

- Example: We want to train 2 topics with α=1 and β=0.1

Initialize topics randomly

Doc 1	a	a	b	b	a	a
	2	2	1	1	2	1
Doc 2	a	b	b	c	c	a
	2	1	1	1	2	2
Doc 3	d	d	a	c	c	
	1	2	1	2	1	
Doc 4	d	c				
	1	1				

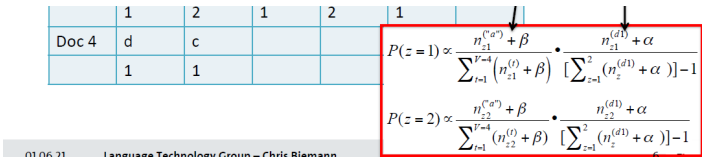
- Example: We want to train 2 topics with α=1 and β=0.1

Sample new topic for first word in first document
→Delete topic of actual word

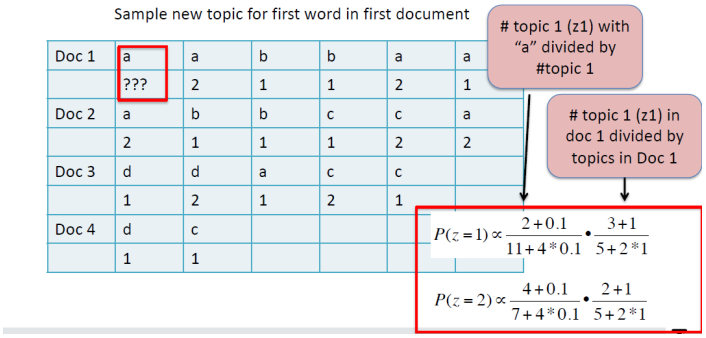
Doc 1	a	a	b	b	a	a
	???	2	1	1	2	1
Doc 2	a	b	b	c	c	a
	2	1	1	1	2	2
Doc 3	d	d	a	c	c	

topic 1 (z1) with "a" divided by #topic 1

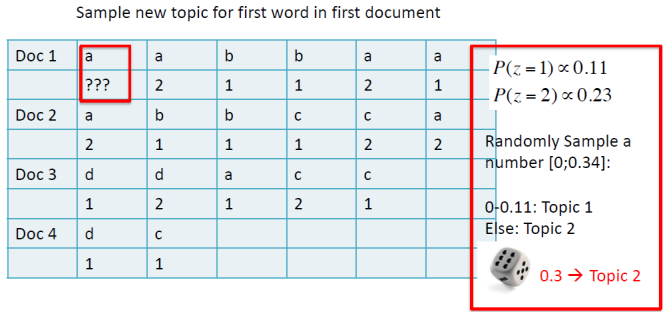
topic 1 (z1) in doc 1 divided by topics in Doc 1



- Example: We want to train 2 topics with α=1 and β=0.1



- Example: We want to train 2 topics with α=1 and β=0.1



- Example: We want to train 2 topics with α=1 and β=0.1

Now we can calculate the word topic distribution

Doc 1	a	a	b	b	a	a
	1	1	1	1	1	1
Doc 2	a	b	b	c	c	a
	1	1	1	2	2	1
Doc 3	d	d	a	c	c	
	2	2	2	2	2	
Doc 4	d	c				
	2	2				

1) Count #topic/term

2) Use Smoothing factor β for zero counts

	Topic 1	Topic 2
a	0.59	0.11
b	0.39	0.01
c	0.01	0.55
d	0.01	0.33

3) Normalize per topic

In [ ]: