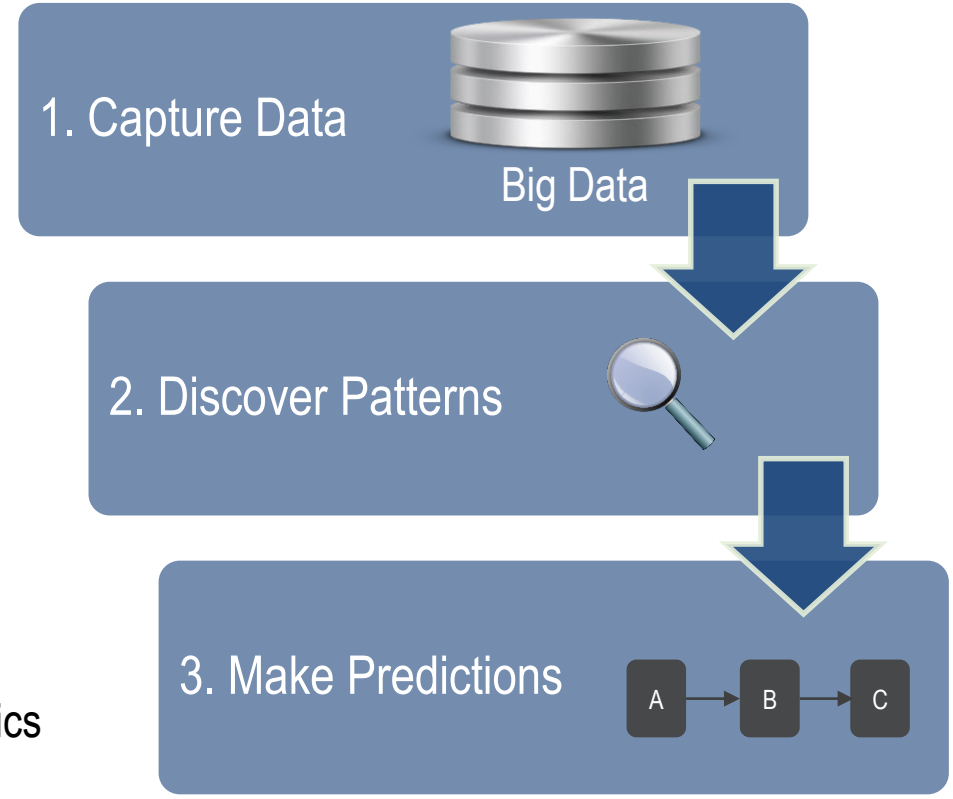
What do we want to do?

Organization‘s Challenge: Generating business value from Big Data

- Analytics
  - Discover meaningful patterns in data
  - Guide decision-making
- Descriptive Analytics
  - Summarize what happened
  - Example: number of Facebook likes
- Predictive Analytics
  - Forecast what might happen
  - Example: regression analysis
- Prescriptive Analytics
  - Oftentimes seen as a subset of predictive analytics
  - Recommend actions



- Predictions about the future are highly valuable for organizations
- Predictive Analytics encompass Prescriptive Analytics

Why R?

- Students today learn R at the university (other statistical programming languages/tools such as SAS, SPSS, SAP PAL are much less taught)
- R is a comprehensive platform
- R is flexible and customizable
- R is free (e.g., SAS is expensive)

Source: Revolution Analytics (2014)

Definition Text Mining

Definition: Text Mining

"Application of data mining to non-structured or less structured text files. It entails the generation of meaningful numerical indices from the unstructured text and then processing these indices using various data mining algorithms large databases."

- Turban et al. (2007)

Key Takeaways:

- Specific application of Data Mining
- Non-structured text
- Generation of indices
- Processing of these indices

Text Mining Software Functionalities

Preprocess

- Data preparation
- Data importing
- Cleaning
- General preprocessing

Associate

- Association analysis
- Finding associations for a given term based on counting and co-occurence frequencies

Cluster

- Clustering of similar documents
- Building groups

Summarize

- Summarization of important concepts in a text
- Identification of high-frequency terms

Categorize

- Classification of text into predefined categories

API

- Availability of application programming interfaces
- Possibility to extend program with plug-ins

Source: Feinerer et al. (2008)

Text Mining

Product	Preprocess	Associate	Cluster	Summarize	Categorize	API
Commercial						
Clearforest	✓	✓	✓	✓		
Copernic **Summarizer**	✓			✓		
dtSearch	✓	✓		✓		
Insightful **Infact**	✓	✓	✓	✓	✓	✓
Inxight	✓	✓	✓	✓	✓	✓
SPSS **Clementine**	✓	✓	✓	✓	✓	
SAS **Text Miner**	✓	✓	✓	✓	✓	
TEMIS	✓	✓	✓	✓	✓	
WordStat	✓	✓	✓	✓	✓	
Open Source						
GATE	✓	✓	✓	✓	✓	✓
RapidMiner	✓	✓	✓	✓	✓	✓
Weka/KEA	✓	✓	✓	✓	✓	✓
R/tm	✓	✓	✓	✓	✓	✓

Table 1: Overview of text mining products and available features. A feature is marked as implemented (denoted as ✓) if the official feature description of each product explicitly lists it.

Source: Feinerer et al. (2008)

R Package „tm“ (R/tm)

A Brief introduction to „R/tm“

- All text documents that shall be used for text mining (for instance, one million Twitter messages) are collected and stored in one object. This object is usually called *corpus*.

  In other words: a corpus represents a database for all texts!

  – Example: R/tm can load very different texts into a corpus: plain text, PDF, Microsoft Word, HTML, various Reuters formats etc.

- Single words of a sentence are usually referred to as *tokens*.
- Texts then are usually represented as *matrices* based without considering tokens.

  – Example: text 1 = „text mining is fun“; text 2 = „a text is a sequence of words“

Text id	a	fun	is	missing	of	sequence	text	words
1	0	1	1	1	0	0	1	0
2	2	0	1	0	1	1	1	1

  – The matrices of R/tm also can be used by other text mining packages for R (e.g., RWeka, OpenNLP)!

Source: Feinerer et al. (2008)

R/tm – Example Text Transformations

Transformation	Description
Plain text	Convert formatted texts to plain texts (e.g., PDF → plain text)
Remove citation	Remove citation from e-mail texts
Remove numbers	Deletes all numbers
Remove punctuation	Remove punctuation marks
Remove signature	Removes signatures from e-mail
Remove very common words	Removes very common words („stopwords“) such as „and“, „is“, „not“, „the“
Replace words	Replaces words with given words or phrases; e.g., synonyms
Stem text	Remove word suffixes (e.g., „engaged“ → „engag“)
Remove empty spaces	Remove extra empty spaces („whitespaces“)
Lower case	Transform all letters to lower case letters
Tag texts	E.g., tag texts with some grammer information (past tense, verb, noun etc.)

Source: Feinerer et al. (2008; 2014)

R/tm – Text Analysis

Count-based evaluation

- Count = terms with the highest occurence are most important
- Basic analysis that allows more complex analysis, e.g., computation of correlation between words or clustering/classification

Text clustering

- Clustering = groups are unknown at the beginning
- Multiple potential similarity measures exist
- Example use cases:
  - Classification of customer reviews
- Example algorithms:
  - Hierarchical clustering
  - K means clustering

Text classification

- Classification = groups are known in the beginning
- Example use cases:
  - Classification of e-mail into regular, phishing, spam
  - Classification of news articles
- Example algorithms:
  - K nearest neighbors classification
  - Support vector machine classification

Clustering with string kernels

- These are not based on term counts! Instead, they are using the texts directly!
- Usually requires additional R libraries!

Source: Feinerer et al. (2008)

Example: Google Autocomplete

Case 2 – Steps

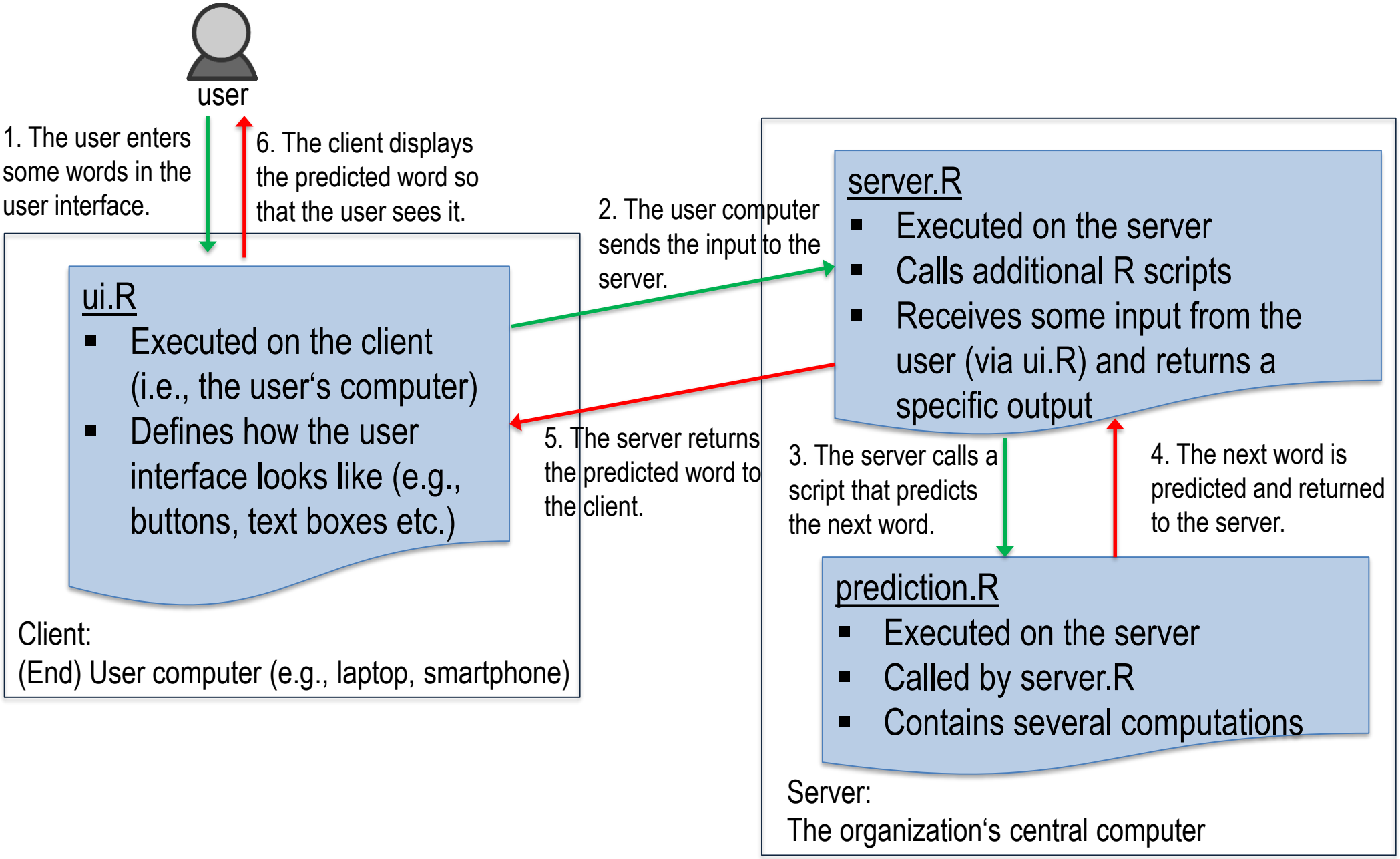
Approach:

1. Create a dataset that contains the word that contains all possible predictions
   - Since real-time analysis is required, all possible predictions shall be computed in advance. Thus, we only need to look up a particular prediction if the user inserts a particular sentence (or part thereof).
2. Create an application which takes some user input and predicts the next word
   - More accurately, the application does not need to predict the next word anymore. It only needs to look up the (already) predicted word.

Step 2 – Used R Libraries

- **Data.table**: a set of high-performance functions for working with data in a tabular format
- **Shiny**: functions for client server management
- **Tm**: functions and data structures for text mining
- **Tau**: additional functions for analyzing texts (text analysis utilities)

Step 2 – R scripts, dataflow and computation



Step 2 – ui.R

```r
# ui.R
library(shiny)

shinyUI(fluidPage(

  titlePanel("Machine Learning Example 2 - NLP"),

  sidebarLayout(

    sidebarPanel(
      #helpText("Predict Next Word"),
      textInput(inputId="ngram", label = "Input Text", value="Enter text here"),
      actionButton("doPrediction", "Predict Next Word")
    ),

    mainPanel(
      p('Predicted next word:'),
      textOutput("result")
    )

  )

))
```

10 lines of code

Step 2 – server.R

```r
# server.R
library(shiny)
library(data.table)
source("./scripts/prediction.R")

bi.dt = data.table( read.csv(file="./data_images/english_bigram.csv", header=TRUE, sep = ";") )
tri.dt = data.table( read.csv(file="./data_images/english_trigram.csv", header=TRUE, sep=";") )

shinyServer(

  function(input, output) { #input=ngram; output=next word

      output$result <- renderText(
      sprintf(predictNextWord(input$ngram, tri.dt, bi.dt))

    )

  }

)
```

9 lines of code

Step 2 – prediction.R

```r
# prediction.R:
library(data.table) # required
library(tm)
library(tau)

predictNextWord <- function(sentence, tri.dt, bi.dt)
{
  sentence.tokens = tokenize(stmt)
  tokens1 = sentence.tokens[sentence.tokens != " "]
  numberOfTokens = length(tokens1)
  a = tokens1[numberOfTokens - 1 - 1]
  b = tokens1[numberOfTokens - 1]
  c = tokens1[numberOfTokens]
  bigram = paste(b, c, sep=" ")
  unigram = a

  predictedWord = "the" #set default predicted word
  temp.dt = NULL

  if( length(tri.dt[tri.dt$ab == bigram,]$c)==1 ){
    # Check whether a trigram of the words a+b+c exists.
    # If so, store it as "predictedWord":
    temp.dt = tri.dt[tri.dt$ab == bigram,]
    temp.dt$c = as.character(temp.dt$c)
    predictedword = temp.dt[[1, "c"]]
    temp.dt = NULL
  }else if( length(bi.dt[bi.dt$b == unigram,]$c)==1 ){
    # If no trigram exists, check whether a bigram of
    # the words b+c exists. If so, store it as "predictedWord":
    temp.dt = bi.dt[bi.dt$b == unigram,]
    temp.dt$c = as.character(temp.dt$c)
    predictedword = temp.dt[[1, "c"]]
    temp.dt = NULL
  }

  return(predword)
}

```

25 lines of code