Statistical Language Models
2019
Week 2 part 1

Dr. Dave Campbell
davecampbell@math.carleton.ca

Approximate Course Outline

Week 1: ShinyApps and Dashboarding

Week 2: TidyText & obtaining data, dealing with time events

Week 3: Regular Expressions; Word co-occurrence explorations

Week 4: Sentiment Analysis; Stochastic process models

Week 5: Exponential models for time between events.

Week 6: Bayesian Basics; Author attribution models; hierarchical models

Week 7: MCMC Diagnostics

Week 8: Embeddings and Word2Vec; Cryptography

Week 9: Clustering; Latent Dirichlet Allocation and topic models.

Week 10: Variational Inference

Week 11: Getting Fancier with Language Models

Week 12: Student projects and presentations

Approximate Course Outline

Week 1: ShinyApps and Dashboarding

Week 2: TidyText & tidy data - obtaining text from Gutenberg, Twitter data - obtaining data from Twitter via API, POSIX time formats

Week 3: Regular Expressions; Word co-occurrence explorations

Week 4: Sentiment Analysis; Stochastic process models for sentiment category; predict the limiting distribution of sentiment categories for excluded chapter, new book, or new news items.

Week 5: Exponential models for time between events. Poisson events and exponential time between events;

Week 6: Bayesian Basics; Author attribution models; hierarchical models

Week 7: MCMC Diagnostics

Week 8: Principal Component Analysis as an embedding. Predicting next word via stochastic process massively sparse. Word2Vec models; Neural Net embeddings; Distributions on a circle; De-biasing Neural Nets; cryptography via dictionary and maximum likelihood likelihood

Week 9: Clustering; Latent Dirichlet Allocation and topic models. Issues of multimodality.

Week 10: Variational Inference

Week 11: Getting Fancier with Language Models, Long Short Term Memory models and

Week 12: Student projects and presentations

More R details

R as a calculator:

38+4

6*7

#R can hold text strings

paste("forty","two")

we can define variables

answer = 42

we can use logical expressions

answer >30

answer ==40

answer <-40

#logic can be used on text:

tip = "don't panic"

but we can look inside the text to see if a pattern exists

grepl(tip, pattern = "panic")

Install libraries once

#Install one library:

install.packages("dplyr")

#Install a few at a time:

install.packages(c("tidytext","gutenbergr","ggplot2","stringr"))

Load libraries every time you use them

```
library(tidytext)

library(gutenbergr)

library(ggplot2)

library(stringr)


#OR:

lapply(c("tidytext","gutenbergr","ggplot2","stringr"),require, character.only = TRUE)
```

^ note **require** instead of **library** here because it is inside a function

Text as data
More MacBeth available from [Gutenberg] {http://www.gutenberg.org/files/1533/1533-0.txt}.

MacBethAct1Scene1 =
c("ACT I

SCENE I. An open Place.

Thunder and Lightning.
Enter three Witches.

FIRST WITCH.

When shall we three meet again?

In thunder, lightning, or in rain?

SECOND WITCH.

When the hurlyburly's done,

When the battle's lost and won.

THIRD WITCH.

That will be ere the set of sun.

FIRST WITCH.

Where the place?

SECOND WITCH.

Upon the heath.

THIRD WITCH.

There to meet with Macbeth.

FIRST WITCH.

I come, Graymalkin!

SECOND WITCH.

Paddock calls.

THIRD WITCH.

Anon.

ALL.

Fair is foul, and foul is fair:

Hover through the fog and filthy air.")

Formatting text

\n and \t

Tokenize: the smallest useful segment of data (Act? Scene? Line? Word?)

MacBethAct1Scene1 = unlist(strsplit(MacBethAct1Scene1,split="\n"))

tibble

A tibble is a data structure that stores all of the information in a way that can be easily retrieved and processed.

data.frame —> matrix of mixed data types but text = factor

Tibble —> matrix of mixed data types but text = text

Useful to track ACT, SCENE, LINE,…

Act1Scene1Lines_df = tibble(line = **1:38**, text = MacBethAct1Scene1, act = 1, scene = 1)

Subdivide the text lines into tokens defined as words, sentences, some arbitrary pattern.

unnest_tokens(Act1Scene1Lines_df,output = word,input = text, token = "words")

Piping & tibbles aka why we use tidyverse

```
Act1Scene1Lines_df %>%
  unnest_tokens(       output = word,input = text, token = "words") # same result but using piping

Act1Scene1Lines_df %>%
  unnest_tokens(   output = sentence,input = text, token = "sentences") # token is a sentence

Act1Scene1Lines_df %>%
  unnest_tokens(     output = ngrams,input = text, token = "ngrams", n = 2) # token is a bigram

Act1Scene1Lines_df %>%
  unnest_tokens(      output = witchy,input = text, token = "regex", pattern = "WITCH.") # token is everytime the WITCH. is mentioned.
```

All of Macbeth

library(gutenbergr)

Macbeth = gutenberg_download(1533) # <—- more about this later

 # Act tokens

acts = Macbeth %>% unnest_tokens(output = scenes,input = text, **token = "regex"**, pattern = "**ACT**")

a cheap way to show the entire (untruncated) text from a tibble is to use one of:

#acts, #View(acts) , #c(acts)

Split the play into Scenes

scenes = Macbeth %>% unnest_tokens(output = scenes,input = text, token = "regex", pattern = "**SCENE**")

Counting occurrences

#Find the counts of the most common words:

```
Macbeth %>%  unnest_tokens(  output = individualwords,input = text, token = "words")  %>%
    count(individualwords,sort = TRUE)
```

#Specifically find just the count for the word "act"

```
Macbeth %>%  unnest_tokens(     output = individualwords,input = text, token = "words")  %>%
  count(individualwords) %>%.  subset(individualwords == "act")
```

Counting a word ≠ counting useful words from context

Detour to regular expressions!

Back to Gutenberg

```
library(gutenbergr)

gutenberg_works()
```

Logical expressions: Find all the Jane Austin books

```
# logical expression uses double equal sign '=='

gutenberg_works(author == "Austen, Jane")

gutenberg_works(author == "Austen")


# stringr detect a specified pattern

gutenberg_works(str_detect(author, "Austen, Jane"))

gutenberg_works(str_detect(author, "Austen"))


#Also find a specific pattern

gutenberg_works(grepl(author,pattern = "Austen, Jane"))

gutenberg_works(grepl(author,pattern = "Austen"))
```

Herbert George Wells

To make sure that we obtain only the author that we want books, let's count the unique authors that were found.

```
gutenberg_works(str_detect(title, fixed("war of the worlds")))
```

```
gutenberg_works(str_detect(title, fixed("war of the worlds",
ignore_case=TRUE)))
```