Text Scraping and LDA

Apoorva Lal, Zuhad Hai

May 14, 2021

Stanford

Text Scraping

Unsupervised Learning on Text

Text Scraping

- wget

```
wget -A '*debates201*' -r -np -nc -l1 --no-check-certificate
  -e robots=off
  https://www.theyworkforyou.com/pwdata/scrapedxml/debates/
```

- curl

How to Get Text (Or Other Data)?

- Scrape from websites
  - use `beautifulSoup` in Python or `rvest` in R
  - easiest if provided data are accessible
  - with large datasets, hard to do (timeout and bandwidth problems)
  - scraping is significantly easier if you can discover regularities in the source data $\rightarrow$ `EXAMPLE (local elections)`

How to Get Text (Or Other Data)? (cont’d)

- Example use case for `rvest`:

```
lego_movie <- read_html(”http://www.imdb.com/title/tt1490017/”)
(rating <- lego_movie %>% html_nodes(”strong span”) %>%
  html_text() %>% as.numeric())
#> [1] 7.8
(cast <- lego_movie %>%
  html_nodes(”#titleCast .primary_photo img”) %>%
  html_attr(”alt”))
cast
#>  [1] ”Will Arnett”     ”Elizabeth Banks” ”Craig Berry”
#>  [4] ”Alison Brie”     ”David Burrows”   ”Anthony Daniels”
#>  [7] ”Charlie Day”     ”Amanda Farinos”  ”Keith Ferguson”
```

How to Get Text (Or Other Data)? (cont'd)

- Scrape from pdfs
    - if text is machine-readable, use `pdftools` or `tabula`
    - if text is not recognised, use OCR software (e.g., `LayoutParser`)
- Bottom line: Original data easy to get once you're familiar with the tools!

Unsupervised Learning on Text

Singular Value Decomposition on Document Term Matrices

- Recall that any matrix can be written as $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$
- Applied to document term matrices, this is called 'latent semantic indexing'
    - $\mathbf{U}$ is known as a 'term matrix'
    - $\mathbf{V}^\top$ is known as a 'document matrix'
- Goal is to obtain a low-dimensional approximation of DTM by zeroing out rows and columns corresponding with smaller eigenvalues [IIR by Manning, Raghavan, Schutze]
- All machine learning is SVD

Latent Dirichlet Allocation

- The idea is to build a hierarchical model to predict probabilities of each document belonging to different clusters.
- We have K topics, M documents, $1, \ldots, i, \ldots, N$ words in each document

$$\underset{1\times K}{\alpha}$$

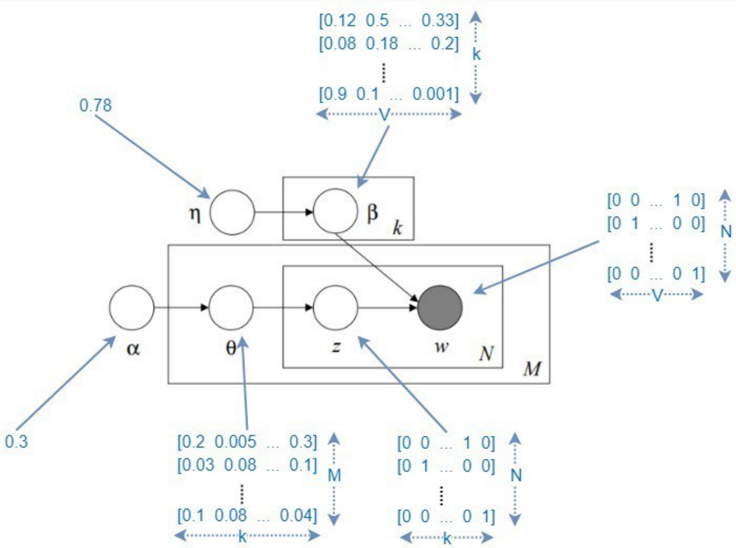
$$\underset{1\times K}{\theta_m} \sim \mathrm{Dir}(\underset{1\times K}{\alpha})$$

$$\underset{1\times K}{z_{im}} | \underset{1\times K}{\theta_m} \sim \mathrm{Multinomial}(\underset{1\times K}{\theta_m})$$

$$\underset{1\times N}{\beta_k} \sim \mathrm{Dir}(\mathbf{1})$$

$$\underset{1\times 1}{x_{im}} | \underset{1\times N}{\beta_k}, z_{imk} = 1 \sim \mathrm{Multinomial}(\underset{1\times N}{\beta_k})$$

LDA in plate notation



Latent Dirichlet Allocation (cont'd)

- θ_m (what Justin calls π_i in his slides) is the vector that describes the probability of a document belonging to each topic.
- β_k (what Justin calls θ_k in his slides) is the vector that describes the probability of word i conditional on topic k.
- We have the generative model – how do we compute these quantities?
  - Joint posterior can be approximated using Gibbs sampling.
  - $\rightarrow$ far deeper dive into material in Doug's 450D (Bayesian statistics)
- The neat feature of LDA is that topics and words are interdependent!

- To the Code! $\longrightarrow$ `EXAMPLE (Brexit LDA)`