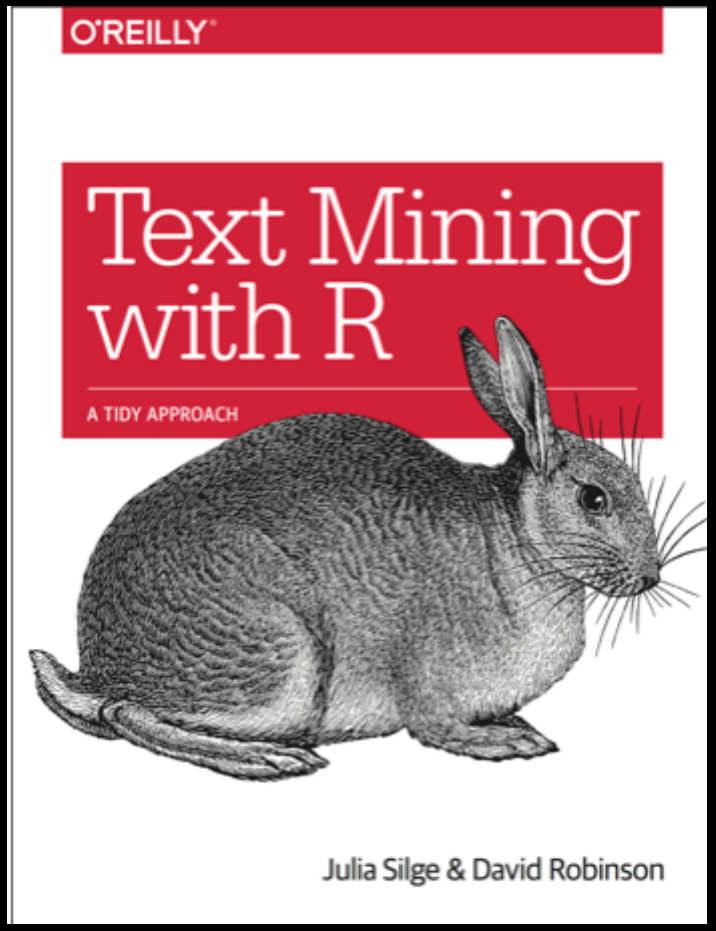
Statistical Language Models
2019
Week 4 part 2

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

https://www.tidytextmining.com/twitter.html



- Great book.
- We've covered Chapters: 1,2,4,7
- Later we'll come back this book for Chapters 4 tf_idf & 6 LDA

- Stochastic process $\{X(t), t \in T\}$, consider discrete case.

  If $X(t) = a$, the stochastic process is at state a at time t

- If, whenever $X(t) = a$, for all t there is a fixed probability P_{ab} that $X(t + 1) = b$, then the stochastic process is a Markov chain

- a Markov chain is a stochastic process where for all states $a_{t-1}, \ldots, a_0$, and for all $t \geq 0$

- $P[X(t+1) = b \mid X(t) = a, X(t-1) = a_{t-1}, \ldots X(0) = a_0) = P_{ab}$

-

- The probability of moving from one state to another is used to make decisions in games. Baseball has been well studies due to the discrete event. Some have even subdivided the events to account for handedness of batters and pitchers.

- See also:

- http://www.pankin.com/markov/theory.htm

- KOOP, Gary, (2004), "Modelling the evolution of distributions: an application to Major League baseball", Journal of the Royal Statistical Society. Series A. Statistics in society, vol. 167 (4), pp. 639-655

- Consider a single player, they can be ’at bat’, on 1st base, 2nd base, 3rd base, ’at home’, or ’out’.
- There are 6 states, and transitions occur between states. Let’s put up some numbers

$$\begin{bmatrix} P_{11} & P_{12} & ... \\ P_{21} & P_{22} & \dots \\ ... & ... & ... \end{bmatrix}$$

- Why are the row sums = 1?

Markov Chain —> only last state matters for prediction

- For a Markov Chain:
  $P_{ij} = P(X(t+1)=j|X(t)=i)$
- $= P\,(X\,(t+1) = j \mid X\,(t) = i\,,\, X\,(t-1) = i1\,,\,\ldots\,,\, X\,(0) = i0\,)$
- History doesn't matter beyond time t. So everytime X = i it is like the chain is restarting along another random path.

Markov Chains

- A Markov Chain is a stochastic process that for any states i and j and any time $t \in T$, if $X(t)=i$,then

- $P\big(X[t+1] = j \mid X[t] = i\big) = P_{ij}$
  The probability is constant. So the basic regression
  $Y(t) = \beta_0 + \beta_1 t + \varepsilon$ can't be a Markov Chain.

- The transition probability P_{ab} is the probability of moving into state b from state a.
- We can make a full transition matrix describing all the possible states and the probabilities of moving between them.

Baby by Justin Bieber

2 sentiment categories

- https://genius.com/Justin-bieber-baby-lyrics
- Stochastic Processes

Lyrics as a tibble —> examine bigrams and word co-occurrence

- library(genius)
- library(igraph)
- library(tidytext)
- Artist = "Justin Bieber"
- Song = "Baby"
- Lyrics = genius_lyrics(artist=Artist ,song=Song)

- lyrics_bigrams = Lyrics %>% unnest_tokens(bigram, lyric, token = "ngrams", n = 2)
- **lyrics_bigramsCount** = lyrics_bigrams %>% count(bigram,sort=TRUE)
- **bigrams_separated** = lyrics_bigramsCount %>% separate(bigram, c("word1", "word2"), sep = " ")
- bigram_graph = bigrams_separated %>% filter(n > 1) %>% graph_from_data_frame()
- library(ggraph) # makes nicer graph plots
- ggraph(bigram_graph, layout = "fr") +
- geom_edge_link() + geom_node_point() + geom_node_text(aes(label = name), vjust = 1, hjust = 1)
- ##########
- a = grid::arrow(type = "closed", length = unit(.15, "inches"))
- ggraph(bigram_graph, layout = "fr") +
- geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
- arrow = a, end_cap = circle(.07, 'inches')) +
- geom_node_point(color = "lightblue", size = 5) +
- geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
- theme_void()

Build a matrix of the vocabulary

- WordsInOrder = Lyrics %>% unnest_tokens(input=lyric, output=word)
- WordsInOrder = WordsInOrder[!is.na(WordsInOrder$word),]
- Vocabulary = unique(WordsInOrder$word)
- #Build a matrix of all the possible word states:
- N = length(Vocabulary)
- Wordtransitions = matrix(0,nrow=N, ncol=N, dimnames=list(Vocabulary,Vocabulary))

#loop over song words to fill in the number of transitions from (rowname) to (colname)

- for(lp in 1: (dim(WordsInOrder)[1] -1)){
- Wordtransitions[WordsInOrder$word[lp], WordsInOrder$word[lp+1]] =
- Wordtransitions[WordsInOrder$word[lp], WordsInOrder$word[lp+1]] +1
- }
- head(Wordtransitions)

Simpler table: Consider a fewer states

- Dimname2use = c("baby","otherword")
- Babymatrix = matrix(0, nrow=length(Dimname2use), ncol=length(Dimname2use), dimnames = list(Dimname2use,Dimname2use))

- for(lp in 1: (length(Dimname2use) -1)){
- for(kl in 1: (length(Dimname2use) -1)){
- Babymatrix[Dimname2use[lp], Dimname2use[kl]] =
- Wordtransitions[Dimname2use[lp], Dimname2use[kl]]
- }}

Simpler table: Consider a fewer states

- other = setdiff(Vocabulary,Dimname2use)
- #To other words:
- for(lp in 1: (length(Dimname2use) -1)){
- Babymatrix[Dimname2use[lp], "otherword"] =
- sum(Wordtransitions[Dimname2use[lp], other])
- }
- #from other words:
- for(lp in 1: (length(Dimname2use) -1)){
- Babymatrix["otherword",Dimname2use[lp]] =
- sum(Wordtransitions[other,Dimname2use[lp]])
- }
- #Other 2 other
- Babymatrix["otherword","otherword"] =
- sum(Wordtransitions[other,other])
- Babymatrix

Normalize:

- PBabymatrix = t(apply(Babymatrix,1,function(x){x /sum(x)}))

- States j is **accessible** from i if $P_{ij}^{n} > 0$ for some n≥0
- If i and j are **accessible** to each other, they **communicate** and we write i ↔j
- States that communicate are in the same **class**. A chain is **irreducible** if there is only 1 class.
- State i is **recurrent** if $\sum_{m=1}^{\infty} P(\text{next time it reaches state i is in m steps}) = 1$
- State i is **transient** if $\sum_{m=1}^{\infty} P(\text{next time it reaches state i is in m steps}) < 1$.

Chapman-Kolmogorov Equations

- The n step transition probabilities $P_{ab}^{n} = P\left\{X(k+n) = b \,|\, X(k) = a\right\}$ is the probability that a process in state a will be in state b in n additional transitions.
- $P_{ab}^{n+m} = \sum_{k} P_{bk}^{n} P_{ka}^{m}$ for all n,m ≥ 0 and all a,b.

- State i has **period** d whenever $P_{ij}^n = 0$ if n is not divisible by d.
- A state with period 1 is **aperiodic**
- If state i is **recurrent**, then it is **positive recurrent** if the expected time until the process returns to i is finite.
- **Positive recurrent, aperiodic states are called ergodic**
- For an irreducible ergodic Markov Chain $\pi_j = \lim_{n \to \infty} P_{ij}^n$ exists and is independent of the starting point i.