Introduction to Text Mining for Sociolinguistic Research

Joseph Roy

2 November 2016

Introduction

For these notes, I discuss how to analyze a corpus (i.e. collection) of documents where each document might represent a separate sociolinguistic interview, separate letter or separate tweet. This data is unstructured– just a collection of words. The purpose of most text mining techniques is to impose order on the collection of words in each text and across the corpus. There are several names these techniques go by: text mining, text analysis, content analysis, text analytics. They are basically the same set of techniques (some with a slightly different vocabulary or history), but the general idea is automated analysis of language data (text) for some purpose (often predicative – e.g. marketing analytics).

Any text analysis, regardless of language, can involve the following steps:

1. Read text files into R. 2.Remove white space, meta-data, punctuation, numbers and non-UTF-8 characters [anything not a lexical item to be included in the analysis] and make everything lower case.
2. Remove **stopwords** .
3. **Stem** the document.

There are a number of options to move through with (1)-(3), but with 4, unless you code a stemming algorithm yourself, you have to use pre-programmed algorithms.

These techniques can take several different approaches to the unstructured data in a corpus. For topic models and these notes, we take a *bag of words* approach that treats each document as an unordered collection of words. There are more advanced techniques that condition their output on the order of the words or syntactic structure. For any of the code discussed here, there are possible alternative packages or approaches. In the last year alone, a number of packages have come online to make data cleaning easier and processing the statistical models more straightforward. For example, *tidytext* , a package in R just developed in 2016, renders much of the code in the references (see the end of these notes) obselte. In a year or two (or next week), the code presented may not work as advertised: doing text analysis of any sort requires staying up to date with new tools and new techniques. I chose R for this workshop due to its wide availability and the large amount of information online and tutorials for performing many different statistical analyses. Many times if there are issues, you can find answers or post questions at stackoverflow.

Read text files into R.

Reading data into R is relatively simple: you have to specify the location of the text files that comprise your corpus. It is easier if they are the only .txt files in the directory you are using. The text files can be cleaned (i.e. only the text that you are interested in) or include meta-data, tags, etc. that need to be removed. This works easiest with a text file format: Rich Text, Word Documents can be read, but you have to mass convert to .txt at some point [or use separate libraries]. This technique assumes all interviews, or documents are in .txt (or equivalent) format (e.g. the tier file format in Elan format).

```
library(tm)
```

```
## Loading required package: NLP
```

```
library(tidytext)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
library(topicmodels)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate
```

```
library(visreg)
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-15. For overview type 'help("mgcv-package")'.
```

```
ICE_CE = Corpus(DirSource(directory = "C:/Users/Joe/Desktop/ICE Canadian Corpus",
                          pattern = "\\.(txt)$"))

tidyCanadian = tidy(ICE_CE)

head(tidyCanadian)
```

```
## # A tibble: 6 × 8
##   author       datetimestamp description heading          id language
##    <lgl>              <dttm>       <lgl>   <lgl>       <chr>    <chr>
```

```
## 1     NA 2016-11-03 11:37:58          NA      NA S1a-001.txt       en
## 2     NA 2016-11-03 11:37:58          NA      NA S1A-002.txt       en
## 3     NA 2016-11-03 11:37:58          NA      NA S1A-003.txt       en
## 4     NA 2016-11-03 11:37:58          NA      NA S1A-004.txt       en
## 5     NA 2016-11-03 11:37:58          NA      NA S1A-005.txt       en
## 6     NA 2016-11-03 11:37:58          NA      NA S1A-006.txt       en
## # ... with 2 more variables: origin <lgl>, text <chr>
```

If you are running this in R Studio, you should see that a *Large VCorpus* object has been created with 500 elements (and is 17.1 MB or so).

The *tidyCanadian* structure is much easier to deal with than the unruly *ICE_CE* VCorpus object from a programming prospective (especially for beginning R programmers). Much of the current online documentation and published "introduction to text analysis" books do not have this package incorporated yet.

Why tidy formatting? Tidy formatting allows for data sets to piped through transformations, new variables added and plotted all while dealing with the data frame structure instead of having to deal with lists and lists of lists.

There are two important elements in the tidy data frame: the id, which is the file name we read in and the text, which is the full text within the file. For the notes, we divide this further into separate lines.

The Canadian ICE Corpus is made up of several genres of language both written and spoken. I chose to focus here on the private spoken language (as it is the closest genre to the typical sociolinguistic interview).

Removing non-lexical items from the data frame.

```
spokenCan= tidyCanadian %>%
  filter(grepl("S1A|a",id)) %>%
  unnest_tokens(line,text, token = "lines") %>%
  group_by(id) %>%
  mutate(linenumber = row_number() )%>%
  ungroup()

spokenCan= spokenCan%>%
  mutate(line = gsub("<.*?>", "", line)) %>%
  mutate(line = gsub(" '", "'", line)) %>%
  mutate(id = gsub("S1A-", "", id)) %>%
  mutate(id = gsub("S1a-", "", id)) %>%
  mutate(id = gsub(".txt", "", id), uniqid = paste(id,"_",linenumber,sep=""))
```

The first set of commands breaks the data into lines by interview (so each line from an interview is treated as a separate document). The length of a document for topic modeling doesn't have to be long which is why you can use these techniques on twitter or similar social media data, for example. The main reason for doing this is to provide a more coherent set of topics (several iterations of topic modelling inform this) – you can skip these lines (and just replace line = with text= in the second data manipulation).

The gsub command that replaced "<.*?>" with "" is important because it removes all the meta-data from actual text in the corpus. If you are transcribing data into a corpus, it is important to use consistent markers for meta data or tags that allow them to be separated from the actual transcribed text. It also important that non-lexical utterances be consistently transcribed (and marked with a separate syntax, if possible so that you can create a regular expression to remove them easily). My personal preference is to use Perl for complicated text cleaning, but any scripting language (including R) you are familiar with is fine for doing this. Many people find python helpful for this as well.

Stop words

Stop words are those frequent words that are generally removed for any kind of text analysis. Linguistically, these are generally function words. The main reason for doing this is that these words are *not* informative for the analysis we'd like to do.

```
stopwords()
```

```
##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
##  [21] "herself"    "it"         "its"        "itself"     "they"
##  [26] "them"       "their"      "theirs"     "themselves" "what"
##  [31] "which"      "who"        "whom"       "this"       "that"
##  [36] "these"      "those"      "am"         "is"         "are"
##  [41] "was"        "were"       "be"         "been"       "being"
##  [46] "have"       "has"        "had"        "having"     "do"
##  [51] "does"       "did"        "doing"      "would"      "should"
##  [56] "could"      "ought"      "i'm"        "you're"     "he's"
##  [61] "she's"      "it's"       "we're"      "they're"    "i've"
##  [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
##  [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
##  [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
##  [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
##  [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
##  [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
##  [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
## [101] "who's"      "what's"     "here's"     "there's"    "when's"
## [106] "where's"    "why's"      "how's"      "a"          "an"
## [111] "the"        "and"        "but"        "if"         "or"
## [116] "because"    "as"         "until"      "while"      "of"
## [121] "at"         "by"         "for"        "with"       "about"
## [126] "against"    "between"    "into"       "through"    "during"
## [131] "before"     "after"      "above"      "below"      "to"
## [136] "from"       "up"         "down"       "in"         "out"
## [141] "on"         "off"        "over"       "under"      "again"
## [146] "further"    "then"       "once"       "here"       "there"
## [151] "when"       "where"      "why"        "how"        "all"
## [156] "any"        "both"       "each"       "few"        "more"
## [161] "most"       "other"      "some"       "such"       "no"
## [166] "nor"        "not"        "only"       "own"        "same"
## [171] "so"         "than"       "too"        "very"
```

```
stopwords(kind="french")
```

```
##   [1] "au"       "aux"      "avec"     "ce"       "ces"      "dans"
##   [7] "de"       "des"      "du"       "elle"     "en"       "et"
##  [13] "eux"      "il"       "je"       "la"       "le"       "leur"
##  [19] "lui"      "ma"       "mais"     "me"       "même"     "mes"
##  [25] "moi"      "mon"      "ne"       "nos"      "notre"    "nous"
##  [31] "on"       "ou"       "par"      "pas"      "pour"     "qu"
##  [37] "que"      "qui"      "sa"       "se"       "ses"      "son"
```

```
##  [43] "sur"      "ta"       "te"       "tes"      "toi"      "ton"
##  [49] "tu"       "un"       "une"      "vos"      "votre"    "vous"
##  [55] "c"        "d"        "j"        "l"        "à"        "m"
##  [61] "n"        "s"        "t"        "y"        "été"      "étée"
##  [67] "étées"    "étés"     "étant"    "suis"     "es"       "est"
##  [73] "sommes"   "êtes"     "sont"     "serai"    "seras"    "sera"
##  [79] "serons"   "serez"    "seront"   "serais"   "serait"   "serions"
##  [85] "seriez"   "seraient" "étais"    "était"    "étions"   "étiez"
##  [91] "étaient"  "fus"      "fut"      "fûmes"    "fûtes"    "furent"
##  [97] "sois"     "soit"     "soyons"   "soyez"    "soient"   "fusse"
## [103] "fusses"   "fût"      "fussions" "fussiez"  "fussent"  "ayant"
## [109] "eu"       "eue"      "eues"     "eus"      "ai"       "as"
## [115] "avons"    "avez"     "ont"      "aurai"    "auras"    "aura"
## [121] "aurons"   "aurez"    "auront"   "aurais"   "aurait"   "aurions"
## [127] "auriez"   "auraient" "avais"    "avait"    "avions"   "aviez"
## [133] "avaient"  "eut"      "eûmes"    "eûtes"    "eurent"   "aie"
## [139] "aies"     "ait"      "ayons"    "ayez"     "aient"    "eusse"
## [145] "eusses"   "eût"      "eussions" "eussiez"  "eussent"  "ceci"
## [151] "cela"     "celà"     "cet"      "cette"    "ici"      "ils"
## [157] "les"      "leurs"    "quel"     "quels"    "quelle"   "quelles"
## [163] "sans"     "soi"
```

Current languages supported: Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Russian, Spanish, Swedish, Catalan, Romanian and smart-english.

The stopwords() function is in the *tm* package, but the *tidytext* package has an extended set of stop words for english (but, not for other languages).

```
stop_words
```

```
## # A tibble: 1,149 × 2
##           word lexicon
##          <chr>   <chr>
## 1            a   SMART
## 2          a's   SMART
## 3         able   SMART
## 4        about   SMART
## 5        above   SMART
## 6    according   SMART
## 7  accordingly   SMART
## 8       across   SMART
## 9     actually   SMART
## 10       after   SMART
## # ... with 1,139 more rows
```

These add low information content words (e.g. actually) that do not help identify topic. For our data, I have added more stop words that were included in the original transcription, but are not content words (or occur so much in every conversation they are revealing).

```
#Word count by unique lexical item

stop2 =c("uh","uhh","yeah", "cos","word","hmm","huh","hm","gonna",
         "mm","word","uh","like","laugh","right","uhm","ya","know",
```

```
         "dont", "ah","yup","duh","stuff","word","words","eacute","qu","wow", "bec",
         "time","sort","lot","wanna","gotta","start","de","ahh", "dunno","people","st","guess")
dfs = data.frame(word=stop2,lexicon="added")
dfs = rbind(dfs,stop_words)

by_interview_word <- spokenCan %>%
  unnest_tokens(word, line) %>%
  anti_join(dfs)
```

```
## Joining, by = "word"
```

Stemming

The purpose of stemming a corpus is to remove morphological endings that are not useful to the final topic models (or other analyses of text). The intuition behind this process is that the words win, winning, winner all are about the same topic (or for any other kind of classification – the lexical stem provides the most information). If for a research question the analyst believes that the morphological endings are informative, they certainly can be left in place before running the topic model.

Many times researchers want to know if they can stem a particular class of items and not others (i.e. verbs only? nouns only?). The short answer is no. The long answer is to find a programmer or computer scientist to collaborate with: there are no, to my knowledge, out of the box tools that do this, but I suspect that it can be programmed (where a part-of-speech tagger is used first and then only part of the data are fed into the stemming algorithm).

To stem the data, which for the final topic models presented here I do not do (run the data with and without stemming to see the difference in the results) we would need to use the stemDocuments function in the tm package.

```
#Word count by unique lexical item

by_interview_word <- by_interview_word %>%
  mutate(line=wordStem(line))
```

Topic Modeling

The formal mathematical term for what we label as topic modeling is *Latent Dirichlet Allocation* and was developed David Blei and colleagues for discovering automatically the topics contained in collection of texts – see (https://www.cs.princeton.edu/~blei/publications.html) for an overview.From Blei (2012: 77) *Topic models are algorithms for discovering the main themes that pervade a large and otherwise unstructured collection of documents.* Topic, for a linguists, can correspond potentially to several different ideas.

Mathematically, we are assuming there are set of topics, latent (i.e. to be discovered and not coded directly) that exists with a set of lexical items distributed within each topic with a given probability (i.e. in LDA in R, this is β) and that each document (or separate text) in your corpus has an probability associated with each topic (in LDA in R this is γ).

Assumes a collection of texts that can be reduced to a set of topics. Topic is sometimes equivalent to *genre* in corpus linguistics or to *sociolinguistic topic* (e.g. School, Work, etc.). In fact, if one were to run a topic model on a corpus sampled by genre (e.g. Penn Parsed Corpus of Early Modern English) this basically recovers the original genres (though in a gradient manner) that were coded for the data. So, you would be able to identify religious texts, but also those texts which are the most irreligious (i.e. comedies).

The interpretation of *topic* is local to the research question and the data. It is straightforward to envision this technique being applied to questions about language, social and cultural attitudes (or orientations). For example, Hoffman and Walker (2010) use a closed answer questionaire and then a PCA to create a ethnic orientation score, but it would be reasonable to ask open-ended questions about the attidude in an area of interest and use topic models to generate a gradient measure from the informants response. Ideally both methods should converge on the same set of results, but the open endeed questions allow for informant response that the researcher may not have envisioned a priori.

There are a couple of ways to approach this when we have the DTM. A more traditional approach would be to do a PCA Principle Component Analysis or Singular Value Decomposition on the DTM matrix. There are also alternative weightings for the DTM that rather than using raw frequency instead put more weight on more informative words (i.e. those that differentiate groups of texts). The tidy text mining book and many other text mining references go into details about this.

First, for topic models, we create the DTM:

```
word_counts = by_interview_word %>%
    count(id, word, sort = TRUE) %>%
    ungroup()

can_dtm = word_counts %>%
    cast_dtm(id, word, n)
```

We then run the LDA function and use the tidy.LDA() function to make a easy to use object of the results.

```
canTopics <- LDA(can_dtm, k = 5, control = list(seed = 1234))
tidyCanTopics <- tidytext:::tidy.LDA(canTopics)
```

We can visualize the results by taking the top 10 terms (or however many) and plotting the probabilities associated with each topic.

```
top_terms <- tidyCanTopics %>% group_by(topic) %>% top_n(10,
    beta) %>% ungroup() %>% arrange(topic, -beta)

top_terms %>% mutate(term = reorder(term, beta)) %>% ggplot(aes(term,
    beta, fill = factor(topic))) + geom_bar(alpha = 0.8, stat = "identity",
    show.legend = FALSE) + facet_wrap(~topic, scales = "free")
```

The results for topics 1 and 8:

Using a Topic Model (LDA) in a regression Analysis

For the results below, I use some results from a topic model built with the Penn-Parsed Corpus of Early Modern English on the spread of do support. As a quick refresher, in Early Modern English DO support, the insertation of inflected DO, in questions and negative declaratives replaced, over a several centuries, the older variant: verb raising. So, the new DO form in (1) replaced the older raised variant in (2).

(1) Do you know John?
(2) Knew you John?

For the results below, I'm going to focus on the topic probalities associated with the first topic that came from the analysis: Relgious texts (where higher probabilities can be intrepreted as the more religious a text

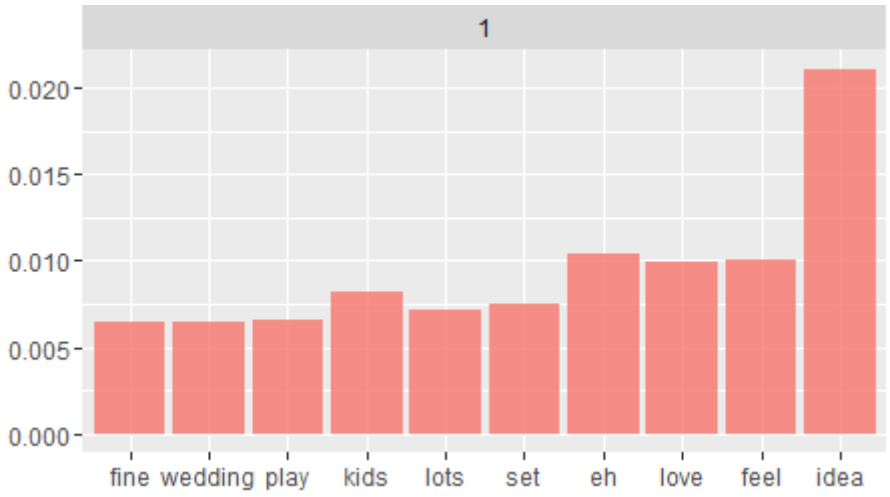


Figure 1: Topic 1

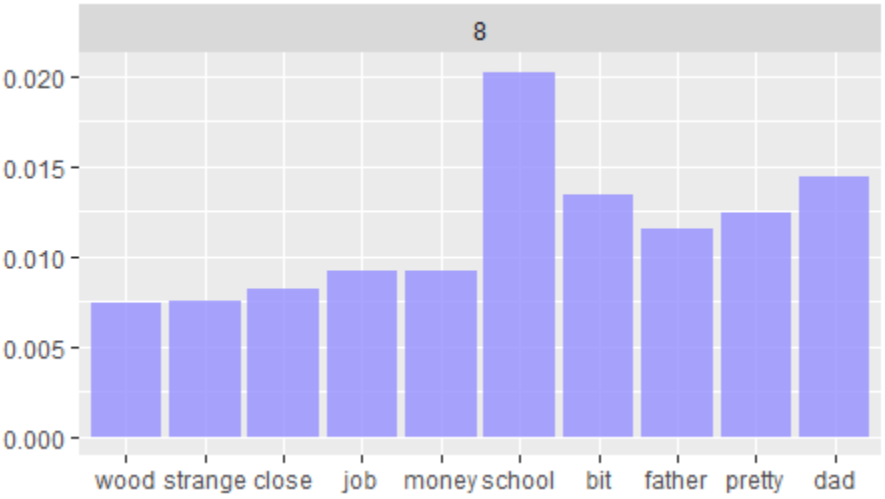


Figure 2: Topic 8

is). The initial step for including a topic probability as a covariate is to merge the topic probabilities with observations from each text. I use the gam() function from the mgcv in order to allow for the relationship between the use of the older variant and topic to be non-linear, if there is evidence in the data for this. Generalized Additive Models provide a way to allow non-linearity via smooths in a regression (see Wood's 2006 book of the same name for an overview – or google it, there are many online tutorials available).

```
tokens = read_excel("auxiliaryData.xlsx")

#To Make my life easier, we need to make all title names into
#lowercase
tokens = mutate_each(tokens, funs(tolower))

#Read excel treats all character data as character vectors (i.e. strings)
#rather than factors. This is actually a good thing, but it means we have
#to explicitly make our factors into factors.

tokens$title = as.factor(tokens$title)
tokens$type = as.factor(tokens$type)

#dir.name = "/data/Dropbox/Workshops/Text Mining (Spring 2016)/week 6/brit"
dir.name = "/home/joe/Dropbox/Workshops/Text Mining (Spring 2016)/week 6/brit"
corp = Corpus(DirSource(directory = dir.name, pattern = "\\.(txt)$"))

IC=corp
IC = tm_map(IC, stripWhitespace)
IC = tm_map(IC, removePunctuation)
IC = tm_map(IC, tolower)
IC = tm_map(IC, PlainTextDocument)
IC = tm_map(IC, removeNumbers)
IC = tm_map(IC, removeWords, stopwords("english"))
IC = tm_map(IC, stemDocument, "english")

#Number of topics
k <- 6

#Latent Dirichlet Allocation (LDA)
ICdtm_tdm_tf <- TermDocumentMatrix(IC, control = list(weighting = weightTf,
                                                      tolower = FALSE))
ICdtm2 <- weightTfIdf(ICdtm_tdm_tf, normalize = TRUE)

#For LDA2
ICdtm_tdm_tf = t(ICdtm_tdm_tf)

))

ldaOut2 <-LDA(ICdtm_tdm_tf,k)
topicProbabilities <- as.data.frame(ldaOut2@gamma)


fnames = gsub("\\.(txt)$","",file.names)
topicProbabilities$title = fnames

#I'm adding six columns, one for each topic, to the
#tokens in my data where the matching column "title" is
```

```
#used to code the data.
tokens2 = merge(tokens, topicProbabilities,by="title")

gam1= gam(I(type=="raising"~s(topic1)+s(topic2)+s(topic3)+s(topic4)+s(topic5)+s(topic6)),
          family="binomial",data=tokens2)
```

In the gam model, the s() represents a fitting a smooth (i.e. allowing for non-linear relationships) to each numeric predictor, the probability associated with each topic for a given text.

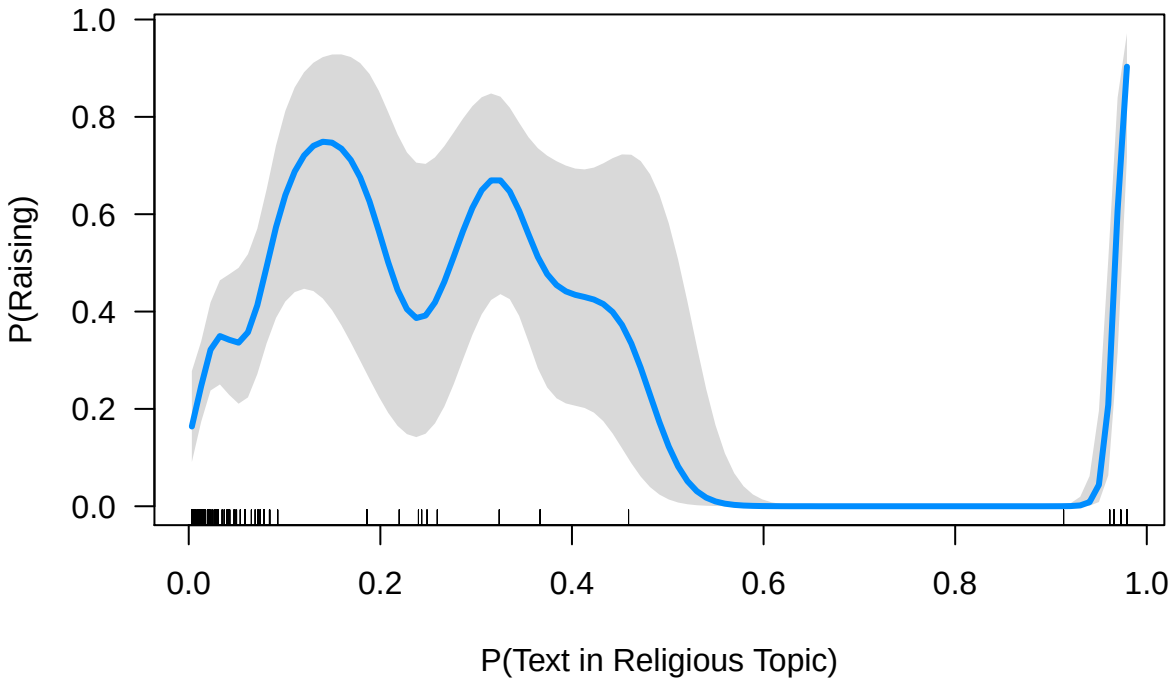
I've simplified the model for pedagogical purposes (when you include other predictors, you have to consider issues of concurvity (non-linear interaction similar to collinearity with regular regression) which is beyond the scope of this workshop).

```
load("gam1.RData")
summary(gam1)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## type == "inversion" ~ s(topict1) + s(topict2) + s(topict3) +
##     s(topict3) + s(topict5) + s(topict6)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.23160    0.08329  -14.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df Chi.sq  p-value
## s(topict1) 8.512  8.883 75.167 7.93e-13 ***
## s(topict2) 1.000  1.000 18.331 1.86e-05 ***
## s(topict3) 1.277  1.484  2.799   0.1842
## s(topict5) 4.115  5.020  7.575   0.1869
## s(topict6) 8.090  8.750 16.550   0.0424 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.284   Deviance explained = 24.5%
## UBRE = -0.070636  Scale est. = 1         n = 1266
```

From the output, we can see that that the smooth for topic 1, the religious topic, obtains statistical signficance. It is at this point we need to discuss the issue of statistical coverage with respect to these topic probabilities: coverage is the distribution of the amount of data over the range of the predictor – do we have enough data at all points of the predictor to trust the estimates of the smooth. In order to see that, we need to visualise the fit of the model with respect to topic 1.

```
visreg(gam1,"topict1",scale="response",ylab="P(Raising)", xlab="P(Text in Religious Topic)")
```



Becuase this is a logistic regression, I can find the fitted probability of using the raised variant (versus DO support) across the values of topic 1. The blue line represents the estimated probability while the grey area represents the predicted 95% confidence intervals on that estimate at a given point. The most important feature of this graph, however, is to show us how topic probability is distributed, or its coverage. On the x-axis, notice the black lines – this represents where you have data points: we can see that there is a lot of data near P(Topic 1) = 0 and as you move acorss the x-axis to 1 there are gaps in coverage. The models still fits a smooth here, but basically it is guessing about the shape of the curve: it is an educated guess based on the statistical properties of these methods, but it still is a guess and, with the low amount of data we have, not a reliable one (in my opinion). We could be justified in coding P(Topic 1/Religious Topic) as a categorical predictor with two or three levels rather than treating it as a contiuous predictor.

Sentiment Analysis

Sentiment in a computer science operationalizes a vague notion of text polarity (positive or negative). This has been refined to more concrete text labeling (i.e. positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust). In the tidytext package, the sentiment data set contains three measures of sentiment. I focus as an introduction to the nrc data in the *sentiment* data set. The sentiment presented below is meant to give an idea of what's available – there are other seniment dictionaries coded with a more fine grained set of scores associated with specific emotions (joy, anger, etc.). Again, we are taking a bag-of-words approach where order doesn't matter which means that the *happy* in *happy* and in *not happy* both are weighted with the same positive score (which could be offset by the score for *not*).

```
wordlist = spokenCan %>%
  unnest_tokens(word,line)
```

```
  wordlist %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 9,679 × 2
##     word     n
##    <chr> <int>
## 1      i  7526
## 2    and  6398
## 3    the  6155
## 4    you  5245
## 5     to  4797
## 6      a  4046
## 7     it  3764
## 8   that  3539
## 9     of  3085
## 10  know  2838
## # ... with 9,669 more rows
```

```
library(tidyr)
bing <- sentiments %>%
  filter(lexicon == "bing") %>%
  select(-score)
```

I can also visualize this data (separate the positive and negative weighted words by how often they occur in the text):

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
wordlist %>%
  inner_join(bing) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                   max.words = 100)
```

```
## Joining, by = "word"
```

negative

positive

For each of the 100 texts in the private spoken genre we now have an estimate of the sentiment for each.

```
spSent <- wordlist %>%
  inner_join(bing) %>%
  count(id, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

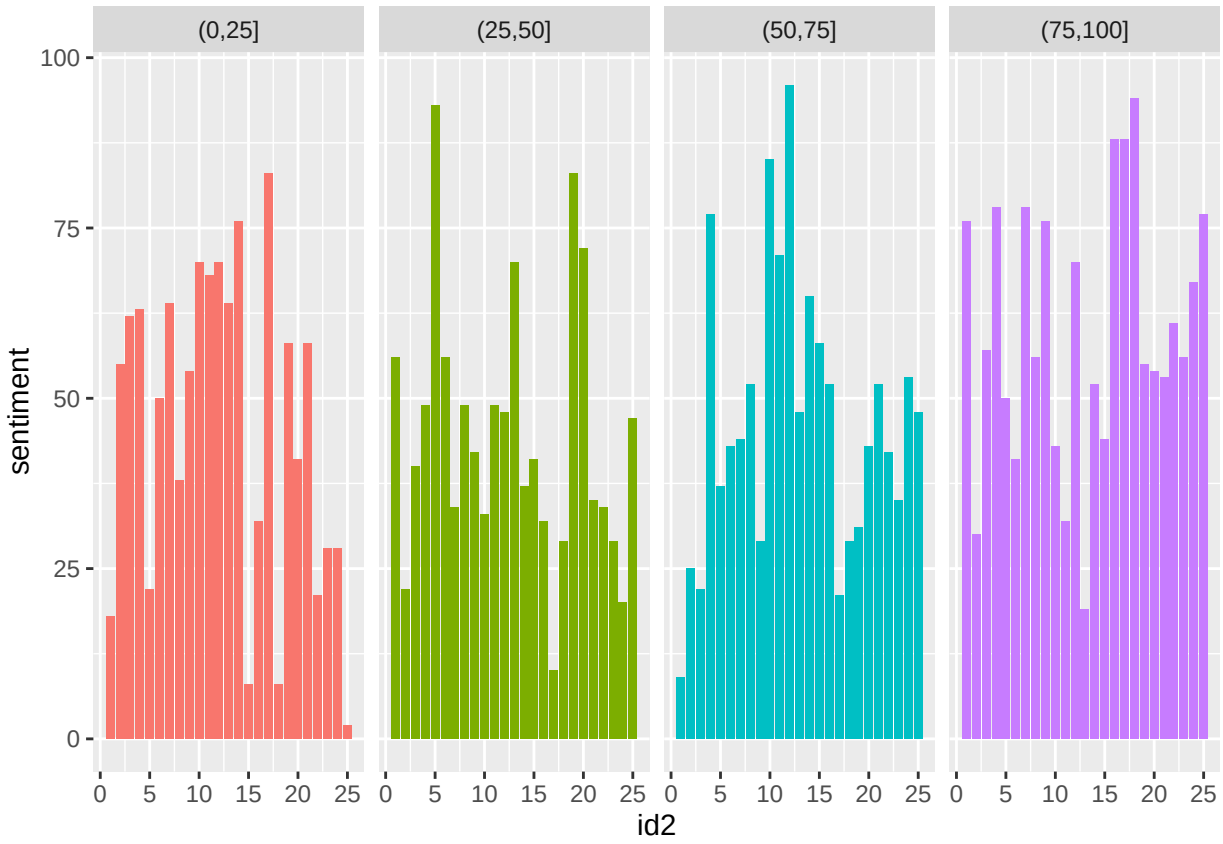
```
## Joining, by = "word"
```

```
head(spSent)
```

```
## Source: local data frame [6 x 4]
## Groups: id [6]
##
##      id negative positive sentiment
##   <chr>    <dbl>    <dbl>     <dbl>
## 1   001       53       71        18
## 2   002       19       74        55
## 3   003       32       94        62
## 4   004       13       76        63
## 5   005       22       44        22
## 6   006       19       69        50
```

We can visualize these results:

```
spSent$id2 = as.numeric(spSent$id)
spSent$group<-cut(spSent$id2,breaks=c(0,25,50,75,100))
spSent$id2 = rep(1:25,times=4)
ggplot(spSent, aes(id2, sentiment, fill=group)) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_grid(~group)
```



Bag of Words Techniques

These techniques pre-suppose a standard and consistent orthographic representation. They also do not account for order, syntactic structure or social structure (e.g. social factor groups). At the end of Blei (2012) he lays out these issues as to be furthered developed, but fortunately over the last four years there are a number of solutions avaialble to researchers (some of which are not widely implemented): dynamic topic models (i.e. with time as a covariate), syntactic topic models (i.e. where the syntactic structure is incorporated into the analysis) and structured topic models (i.e. where predictors can be included in the analysis) have all been developed – many are listed on Blei's homepage and interested researchers can find R implementation for many of the techniques.

Reseacher degrees of freedom

The process of fitting topic models, measuring sentiment and most text mining analyses requires some refinement on the part of the researcher. This of course introduces *reasearcher* degrees of freedom: choices

researchers make along the way that influence the outcome any statistical results have the same impact on the statistical analyses as multiple comparisons (see the work of Andrew Gelman and his notion of *forking paths*). If you use topic modeling and are concerned about accounting for all the scientifically valid alternatives, very recent work on *multiverse analysis* could be of use: Steegen, Sara, Francis Tuerlinckx, Andrew Gelman, and Wolf Vanpaemel. "Increasing Transparency Through a Multiverse Analysis." Perspectives on Psychological Science 11, no. 5 (2016): 702-712.

References

Introduction to Text Mining (Books)

1. Tidy Text Mining with R by Julia Silge and David Robinson http://tidytextmining.com/
2. Miner, Gary. Practical text mining and statistical analysis for non-structured text data applications. Academic Press, 2012.
3. Jockers, Matthew. Text Analysis with R for Students of Literature. Springer, 2014.

Scholarly Articles on the Methodology of Topic Modelling

1. Chang, Jonathan, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. "Reading tea leaves: How humans interpret topic models." In Advances in neural information processing systems, pp. 288-296. 2009.
2. Ignatow, Gabe. "Theoretical foundations for digital text analysis." Journal for the Theory of Social Behaviour (2015).
3. Lee, Monica, and John Levi Martin. "Coding, counting and cultural cartography." American Journal of Cultural Sociology 3, no. 1 (2015): 1-33.
4. Mohr, John W., and Petko Bogdanov. "Introduction-Topic models: What they are and why they matter." Poetics 41, no. 6 (2013): 545-569.

Scholarly Articles that Use Topic Modelling or Text Mining

1. Bail, Christopher A. "The cultural environment: Measuring culture with big data." Theory and Society 43, no. 3-4 (2014): 465-482.
2. DiMaggio, Paul, Manish Nag, and David Blei. "Exploiting affinities between topic modeling and the sociological perspective on culture: Application to newspaper coverage of US government arts funding." Poetics 41, no. 6 (2013): 570-606.
3. Fligstein, Neil, Jonah S. Brundage, and Michael Schultz. "Why the Federal Reserve Failed to See the Financial Crisis of 2008: The Role of"Macroeconomics" as a Sense making and Cultural Frame." (2014).
4. Gentzkow, Matthew, and Jesse M. Shapiro. "What drives media slant? Evidence from US daily newspapers." Econometrica 78, no. 1 (2010): 35-71.
5. Grimmer, Justin. "Appropriators not position takers: The distorting effects of electoral incentives on congressional representation." American Journal of Political Science 57, no. 3 (2013): 624-642.
6. Hargittai, Eszter. "Is bigger always better? Potential biases of big data derived from social network sites." The ANNALS of the American Academy of Political and Social Science 659, no. 1 (2015): 63-76.
7. Jacobi, Carina, Wouter van Atteveldt, and Kasper Welbers. "Quantitative analysis of large amounts of journalistic texts using topic modelling." Digital Journalism 4, no. 1 (2016): 89-106.
8. Jelveh, Zubin, Bruce Kogut, and Suresh Naidu. "Detecting Latent Ideology in Expert Text: Evidence From Academic Papers in Economics." In EMNLP, pp. 1804-1809. 2014.
9. Leskovec, Jure, Lars Backstrom, and Jon Kleinberg. "Meme-tracking and the dynamics of the news cycle." In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 497-506. ACM, 2009.

10. Loughran, Tim, and Bill McDonald. "When is a Liability not a Liability?." Journal of Finance, forthcoming (2009).
11. Mohr, John W., Robin Wagner-Pacifici, Ronald L. Breiger, and Petko Bogdanov. "Graphing the grammar of motives in National Security Strategies: Cultural interpretation, automated text analysis and the drama of global politics." Poetics 41, no. 6 (2013): 670-700.
12. Niculae, Vlad, Srijan Kumar, Jordan Boyd-Graber, and Cristian Danescu-Niculescu-Mizil. "Linguistic Harbingers of Betrayal: A Case Study on an Online Strategy Game." arXiv preprint arXiv:1506.04744 (2015).
13. Resnik, Philip, Anderson Garron, and Rebecca Resnik. "Using topic modeling to improve prediction of neuroticism and depression." In Proceedings of the 2013 Conference on Empirical Methods in Natural, pp. 1348-1353. Association for Computational Linguistics}, 2013.
14. Saavedra, Serguei, Kathleen Hagerty, and Brian Uzzi. "Synchronicity, instant messaging, and performance among financial traders." Proceedings of the National Academy of Sciences 108, no. 13 (2011): 5296-5301.
15. Yu, Dian, Yulia Tyshchuk, Heng Ji, and W. A. Wallace. "Detecting deceptive groups using conversations and network analysis." In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL, pp. 26-31. 2015.

Articles on the Mathematics of LDA

1. Blei D.M., Ng A.Y., Jordan M.I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 993-1022.
2. Phan X.H., Nguyen L.M., Horguchi S. (2008). Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. In Proceedings of the 17th International World Wide Web Conference (WWW 2008), pages 91-100, Beijing, China.
3. Lu, B., Ott, M., Cardie, C., Tsou, B.K. (2011). Multi-aspect Sentiment Analysis with Topic Models. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops, pages 81-88.