* [Text mining infrastucture in R](http://www.jstatsoft.org/v25/i05/" \t "_blank)
* [CRAN Task View: Natural Language Processing](http://cran.r-project.org/web/views/NaturalLanguageProcessing.html)
* [Videos](https://www.youtube.com/user/OpenCourseOnline/search?query=NLP) and [Slides](https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html) from Stanford Natural Language Processing cour
* [Natural language processing Wikipedia page](https://en.wikipedia.org/wiki/Natural_language_processing)

## Course Tasks

This course will be separated into 8 different tasks that cover the range of activities encountered by a practicing data scientist. They mirror many of the skills you have developed in the data science specialization. The tasks are:

1. Understanding the problem
   * Obtain data from course website
   * Familiarize with background
2. Data acquisition and cleaning
   * Tokenization (punctuation, digits, typos)
   * Profanity cleaning
   * Python script used here. Profanity cleansing is suspect.
   * Fixed R script by just taking random 10,000 samples from each
   * *dictCorpus* is catching the corpus before stemming, to be used in the tokenizer script. When refactoring, I’ll just move that part first.
3. Exploratory analysis
   * Making the ngrams … the script I looked at took the top 10, I’m not sure why
   * So, I removed that crap, and now I started tokenizer(smallcorpus, 1) at 2:52 PM … how long will this take?10 minutes. It now has 50,000 entries instead of 10
   * Five most popular words are *the, to, of, and,* and *of.*
   * Digram started at 3:04 pm. This is recursive, holy shit. I hope this doesn’t run in n^3 time. I should have figured out how to get this to use the previous input.
   * 13 minutes for digram, 420,547 observations, this isn’t bad.
   * Trigram started at 3:18, excuse for another beer. Outer Range brewing, the oatmeal cream stout was good, now time for the “High and Dry” Saison.
   * 3:29: This is scaling well. The saison is good
   * Fourgram started at 3:28. Enjoying the saison
   * 3:38. How did this run faster, 10 minutes
4. Statistical modeling
   * The approach will be to find the possible 4grams, withcounts, then the 3 gram corresponding, for the denominator, and calculate up the odds of each word. Then look up the 3 grams of the last 2 words, and similarly find the odds, then look up the onegram of that word, find the odds of the 2grams.
   * Order for this should be to look up the onegram, twogram and threegram, then use those to inform whether to look through the twograms, threegrams and fourgrams, respectively.
   * Need to figure out how to do the stem thing … done
   * So substringing the entire database is kind of slow, that needs to be done in pre-proessing
   * So model here
     1. Look up last word
     2. If last word exists, look up input twogram
     3. If input twogram exists, look up threegram
     4. If input threegram exist, look up possible fourgrams
     5. If fourgrams exist, compute odds of last word of fourgrams and add to outputlist
     6. If input twogram exists, look up possible threegrams
     7. If threegrams exist, get odds of those threegrams
     8. If lastword exists, look up possible twograms
     9. Compute odds of those twograms and add to list
     10. If list is empty, return a WTF are you saying,,
     11. Otherwise order list by odds
5. Predictive modeling
6. Creative exploration
7. Creating a data product
8. Creating a short slide deck pitching your product

# Quiz 1 Notes

Twitter:

[1] maxLength 213

[1] numLines 2360148

[1] hasHate 22138

[1] hasLove 90956

[1] hasPhrase 3

News:

[1] maxLength 5760

[1] numLines 77259

[1] hasHate 322

[1] hasLove 1105

Blogs:

[1] maxLength 40835

[1] numLines 899288

[1] hasHate 11098

[1] hasLove 49167