## Bot or Not Assignment

Applied Computing Assignment: 1 point Due: Start of class **This Thursday** December 1.

For this assignment, you'll be building and analyzing the computational part of a bot by automatically generating text; then you'll be trying to tell apart bot-created text from human-generated text. This involves doing two things drawing on core skills we have emphasized this class.

1. Thinking creatively about sources that are the *input* to computation.
2. Thinking critically about sources that are the *output* of computation.

### Steps

We'll be using a Markov chain to write these bots.

### Making

1. **Choose a source** as input to the Markov chain; download it, and save it as a text file.
2. Run it through the Markov chain generator several times until you get a sentence or two you really like.
3. Choose *both* a bot-like snippet from your original text, and a non-bot-like snippet you've generated, and submit them via the form at https://docs.google.com/forms/d/e/1FAIpQLSdN8y2ONJ9ML6CIaLM5bIgE01nArwVtBjUsbbI4pEg/viewform. You do not need to use an entire snippet generated; you can start at word 5 and end at word 28 of a 40 word run, for instance.

### Reading

Once the bots are up, I'll put a website where you can guess *bot or not*; we'll collect statistics on which ones do the best.

### Grading

You'll be graded on three criteria. Two are objective/competitive; the third is

1. How well your pair of snippets does at confusing fellow students.
2. How well *you* do at telling apart bots from nots.
3. How interesting or challenging the text source you chose was, and/or the snippets produced.

### Appendix 1: Markov Generator Generator code.

This is a function that creates a Markov chain generator from a text file.

```
markov_generator_generator = function(filename, n) {
  library(tidytext)
  library(tidyverse)

  labels = paste("word",1:n,sep="")
```

```
  ngrams =
    read_table(filename,col_names="text")%>%
    filter(!is.na(text)) %>%
    unnest_tokens(ngram,text,token="ngrams",n=n) %>%
    separate(ngram,labels," ") %>%
    group_by_(.dots=labels) %>%
    summarize(count=n()) %>%
    ungroup

  next_word = function(seed) {
    tail = rev(seed)[1:(n-1)]
    names(tail) = paste0("word",(n-1):1)
    tail = as.data.frame(as.list(tail),stringsAsFactors=F)
    tail = tail[,!is.na(tail[1,]),drop=F]
    next_up = ngrams %>%
      inner_join(tail) %>% suppressMessages %>%
      sample_n(1, weight=count) %>%
      select(n) %>% unlist %>% unname
    return(c(seed,next_up))
  }

  generate_list = function(seed,n=50) {
    if (!is.character(seed)) {stop("You must supply a vector of words")}
    seed = tolower(seed)
    for (i in 1:n) {
      seed = next_word(seed)
    }
    seed
  }
}
```

To use it, you'll have to change just a few things: in the line below, both the filename (don't use Malcolm X!) and the depth (n=3 here means to create a level three Markov chain; use the last two words to predict the next 1 word.

```
generator = markov_generator_generator("http://benschmidt.org/X.txt",n = 3)
```

Then you run it.

`set.seed` makes the randomness deterministic; you'll have to change the number for every run. `The` is just a random starting word; you can change it to anything else, and make it a longer list if you like.

```
set.seed(31)
generator(c("the"),24)
```