# Task 3 - Modeling

Help Center

The goal here is to build your first simple model for the relationship between words. This is the first step in building a predictive text mining application. You will explore simple models and discover more complicated modeling techniques.

*Tasks to accomplish*

1. Build basic n-gram model - using the exploratory analysis you performed, build a basic n-gram model for predicting the next word based on the previous 1, 2, or 3 words.
2. Build a model to handle unseen n-grams - in some cases people will want to type a combination of words that does not appear in the corpora. Build a model to handle cases where a particular n-gram isn't observed.

*Questions to consider*

1. How can you efficiently store an n-gram model (think Markov Chains)?
2. How can you use the knowledge about word frequencies to make your model smaller and more efficient?
3. How many parameters do you need (i.e. how big is n in your n-gram model)?
4. Can you think of simple ways to "smooth" the probabilities (think about giving all n-grams a non-zero probability even if they aren't observed in the data) ?
5. How do you evaluate whether your model is any good?
6. How can you use backoff models to estimate the probability of unobserved n-grams?

*Hints, tips, and tricks*

As you develop your prediction model, two key aspects that you will have to keep in mind are the size and runtime of the algorithm. These are defined as:

1. **Size**: the amount of memory (physical RAM) required to run the model in R
2. **Runtime**: The amount of time the algorithm takes to make a prediction given the acceptable input

Your goal for this prediction model is to minimize both the size and runtime of the model in order to provide a reasonable experience to the user.

Keep in mind that currently available predictive text models can run on mobile phones, which typically have limited memory and processing power compared to desktop computers. Therefore, you should consider very carefully (1) how much memory is being used by the objects in your workspace; and (2) how much time it is taking to run your model. Ultimately, your model will need to run in a Shiny app that runs on the shinyapps.io server.

*Tips, tricks, and hints*

Here are a few tools that may be of use to you as you work on their algorithm:

- `object.size()`: this function reports the number of bytes that an R object occupies in memory
- `Rprof()`: this function runs the profiler in R that can be used to determine where bottlenecks in your

function may exist. The `profr` package (available on CRAN) provides some additional tools for visualizing and summarizing profiling data.

- `gc()` : this function runs the garbage collector to retrieve unused RAM for R. In the process it tells you how much memory is currently being used by R.

There will likely be a tradeoff that you have to make in between size and runtime. For example, an algorithm that requires a lot of memory, may run faster, while a slower algorithm may require less memory. You will have to find the right balance between the two in order to provide a good experience to the user.

Created Mon 25 Aug 2014 10:15 AM PDT

Last Modified Wed 29 Oct 2014 5:12 AM PDT