# Final_Project: Markov models for language

*Ling Tan, Xinlu Zhang*

*11/27/2019*

## Introduction

With the development of science and technology, more and more natural sciences can be explained by statistical probability. Researchers believe that the combination of big data and probability can effectively help people in various area in their lives. For example, natural language processing,as a subfield of linguistics, computer science and information science, becomes one of the most signficant and popular area which perfectly combines the real world data with probability model-Hidden Markov model. Artificial Intelligence researchers focus on the processing and understanding human language by imepmenting probability theory to analyze large amounts of natural language data. As far as we can see, understanding language by machine has played a pivotal role in people's common life in the form of translation, extarction, and artifical intelligence personal assitants.

In this project, we focus on learning statistical language models which is the type of models that assign probabilities to the sequences of words, implementing the one-gram and bi-gram Markov models, and applying to 21 English language data sets which about novels written by Charles Dickens. The propose of this project is generating new text from the fitted Markov models.

## Dataset description

The data set we used in this project is the dataset from the first homework, which provided by Prof.Julia Fukuyama. This material was about novels of Charles John Huffam Dickens, who an famous English writer and social critic. The plot in the below shows all the titles of 21 books, number of words in each text martial, number of characters in each text file and number of roles mentioned in each file.

## N-grams

### Conditional probability

The basic idea of n-gram is applying conditional probabilityof a word w given some history text $P(w|h)$. For Example, Suppose the history text h is "there are some master students in this course" and we want to know the probability that the next word is the "which", the probaility function would be represented as

$$P(\text{which}|\text{there are some master students in this course}) =$$
$$\frac{P(\text{there are some master students in this course which})}{P(\text{there are some master students in this course})}.$$

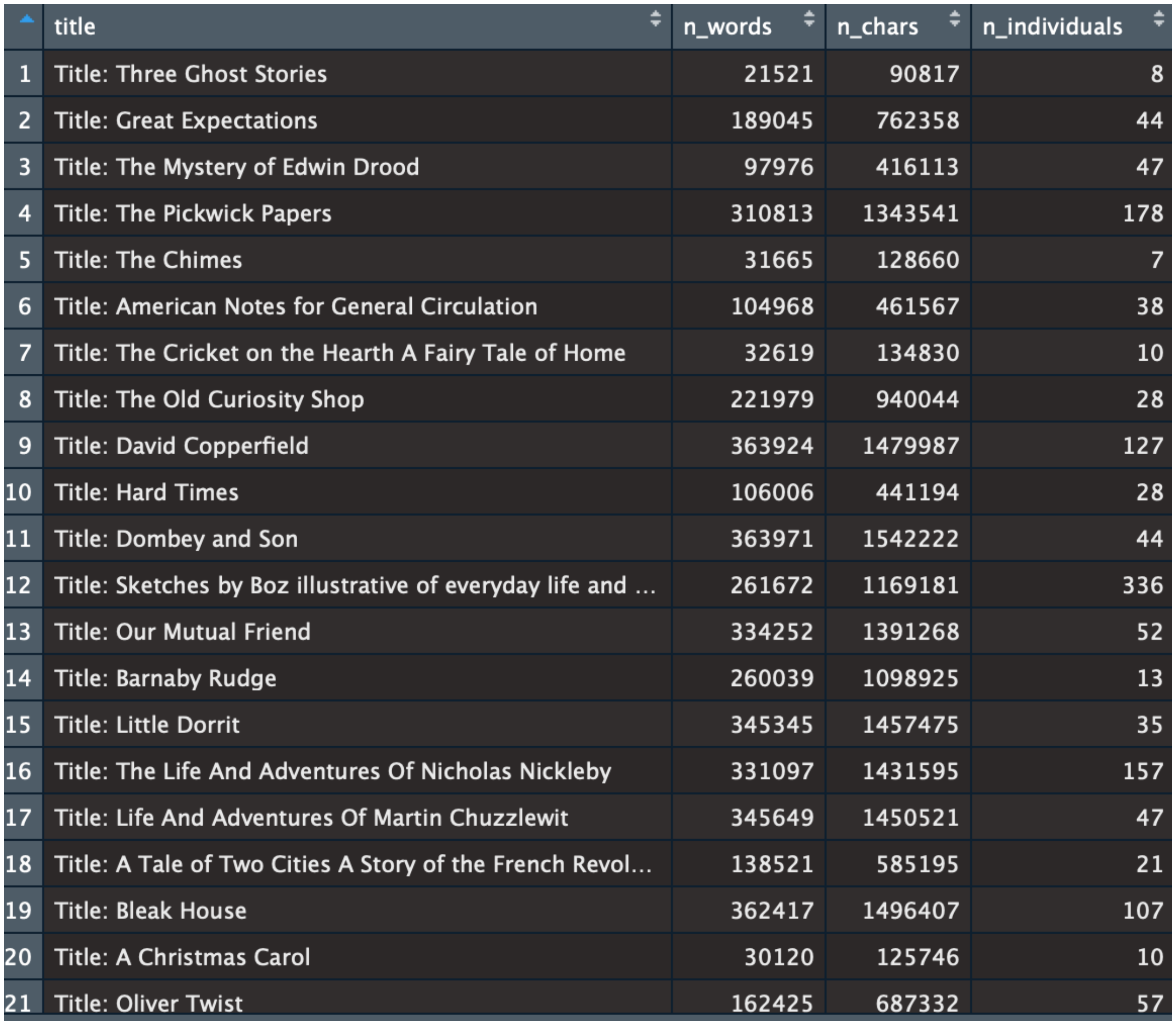
As the large enough number of data set we collect, we can compute the count of these event happened and calculate the probability in the above.

We could also implement the joint probability to resolve the same question, such that $P$(X=there, Y=are, Z=some, A=master, B
$P$(X=there)$P$(Y=are|X=there)$P$(Z=some|Y=are,X=there)$\cdots P$( F=which|X=there, Y=are, Z=some, A=master, B=stude
In general, the chain rule would be,

$$P(X_1 \dots X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1^2)\cdots P(X_n|X_1^{n-1})$$

where the $X_1^{n-1}$ represents the random variable $X_1, X_2, \dots, X_n$.Apply the chain rule to words,

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\cdots P(X_n|X_1^{n-1}) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$



|  | title | n_words | n_chars | n_individuals |
|---|---|---|---|---|
| 1 | Title: Three Ghost Stories | 21521 | 90817 | 8 |
| 2 | Title: Great Expectations | 189045 | 762358 | 44 |
| 3 | Title: The Mystery of Edwin Drood | 97976 | 416113 | 47 |
| 4 | Title: The Pickwick Papers | 310813 | 1343541 | 178 |
| 5 | Title: The Chimes | 31665 | 128660 | 7 |
| 6 | Title: American Notes for General Circulation | 104968 | 461567 | 38 |
| 7 | Title: The Cricket on the Hearth A Fairy Tale of Home | 32619 | 134830 | 10 |
| 8 | Title: The Old Curiosity Shop | 221979 | 940044 | 28 |
| 9 | Title: David Copperfield | 363924 | 1479987 | 127 |
| 10 | Title: Hard Times | 106006 | 441194 | 28 |
| 11 | Title: Dombey and Son | 363971 | 1542222 | 44 |
| 12 | Title: Sketches by Boz illustrative of everyday life and … | 261672 | 1169181 | 336 |
| 13 | Title: Our Mutual Friend | 334252 | 1391268 | 52 |
| 14 | Title: Barnaby Rudge | 260039 | 1098925 | 13 |
| 15 | Title: Little Dorrit | 345345 | 1457475 | 35 |
| 16 | Title: The Life And Adventures Of Nicholas Nickleby | 331097 | 1431595 | 157 |
| 17 | Title: Life And Adventures Of Martin Chuzzlewit | 345649 | 1450521 | 47 |
| 18 | Title: A Tale of Two Cities A Story of the French Revol… | 138521 | 585195 | 21 |
| 19 | Title: Bleak House | 362417 | 1496407 | 107 |
| 20 | Title: A Christmas Carol | 30120 | 125746 | 10 |
| 21 | Title: Oliver Twist | 162425 | 687332 | 57 |

Figure 1: A caption

The chain rule could provide a link relationship between computing the joint probability of a certain word sequence and computing the corresponding conditional probability of a word given previous words.However, it is really hard to compute the exact probability of a word when there is a long words sequence before it, because human language is very creative and any particular context may never happen before.

Instead of calculating $P(w_n|w_1^{n-1})$, researcher foucs on an approximated prbability which only depends on the sequence history by just last few words before the particular words we interested in. For example, if we just want to consider the last word before "which" in the sentence "there are some master students in this course which".Instead of computing the probability of

$$P(\text{which}|\text{there are some master students in this course}),$$

we would like to approximate it by the probability

$$P(\text{which}|\text{course}).$$

## Model implememntation

We say that we could implememnt n-gram models instead of compute the probability of all pervious elements happened. In here, the "$n$" represents the number of words we consider. When $n = 1$, we only calculate the probability of each words appearing which do not depend pn any pervious words at all. For example, instead of joint probability $P(\text{which}|\text{there are some master students in this course})$,,the one gram model, which is always called as unigram model,calculates $P(\text{which})$. In addition, $P(\text{which}|\text{course})$ comes from a bigram model.

### Unigram

A unigram model (one gram model) can be treated as the combination of several one-state finite automaton. It divides the probabilities of separate terms in a context, For instant, instead of calculate joint probability of $ P(t__1, t__2, t__3)= P(t__1) P(t__2| t__1)P( t__3 | t__1, t__2)$, in a unigram model, we focus on $P_{uni}(t_1, t_2, t_3) = P(t_1)P(t_2)P(t_3)$.

In this uni-gram model, the probability of each word only depends on that word's own probability in the particular text document or sequence. Because of that, the unit in this model is only one-state finite automaton. This automaton has a probability distribution which contains all the words in a sequence, and the summing of all possibkle outcome equals to 1. \ In the following code, we programming the unit gram model for all the 21 text documents of Charles Dickens. Even though different text file has it own pattern, we believe building one uni-gram could is better to exlpre the pattern of writing style of Charles Dickens, because a person would have the similar language pattern in his writting.

```
 ##working directory
#opts_knit$set(root.dir = '/Users/zhangxinlu/Documents/IUcourse/stat-s610/hw1/books')
booklist = list.files("/Users/zhangxinlu/Documents/stat610-final/stat610-final")
#booklist = list.files("/Users/addytan/Downloads/books")
textdata<-lapply(booklist, function(x) tm::PlainTextDocument(readr::read_lines(file = x,
    progress = interactive()), id = basename(tempfile()),
    language = "en", description = "Book Filse"))
stop_words<-T
textdata <- tau::textcnt(
  if(stop_words==T) {textdata=gsub("[^[:alnum:] ]", "",textdata)
  tm::removeWords(tm::scan_tokenizer(textdata), tm::stopwords("english"))}
  else {
    tm::scan_tokenizer(textdata)
  }, split = "[[:space:][:punct:][:digit:]]+",method = "string", n = 1L, lower = 1L)
textdata <- plyr::ldply(textdata, data.frame)
colnames(textdata)<-c("word", "frequency")
textdata$text_percent <- textdata$frequency / sum(textdata$frequency)*100
new_onegrame_textdata <- textdata[order(-textdata$frequency),]
```

We show the top 10 words out of 34254 words in text documents. According to the two frequency table, we can see that, "I" and "Mr" have really high frequency than other words, which indeicates that Charles Dickens's novels may pay much attention on the first-person perspective and male.

```
head(new_onegrame_textdata,n=10)
```

```
##         word frequency text_percent
## 19795      i     78306    3.0905168
## 26396     mr     30889    1.2191016
## 34361   said     29776    1.1751747
## 44864      x     25605    1.0105571
## 39271    the     14088    0.5560136
## 28281    one     12768    0.5039169
## 41948   upon     12039    0.4751453
## 23801 little     11226    0.4430585
## 18526     he      9442    0.3726491
## 26405    mrs      9204    0.3632559
```

### Bigram

A bigram model is Aan n-gram model for n=2, which represent a model is built by a sequence of two adjacent elements from a string of tokens. The frequency distribution of bigram model in a squence is one of the most commonly used model for siample statsitical analysis of text data set in various applications, such as speech recognition and search engines recognition.

The basic idea of bigrams is implementing the conditional probability of a token given the preceding token, which could be represent inj mathematics formula:

$$P(W_n|W_{n-1}) = \frac{P(W_{n-1}, W_n)}{P(W_{n-1})}.$$

That is, the probability of the token word $W_n$ given the preceding token word $W_{n-1}$ equals to the probability of its bigram. Or, it can also be calculated by the co-occurance of probability of two token words $P(W_n, W_{n-1})$ divede by the probability of pervious token word $P(W_{n-1})$.

In the following section, we implemented the bigram model and applied to the Charles Dickens's text documents.

```
bigram_textdata<-lapply(booklist, function(x) tm::PlainTextDocument(readr::read_lines(file = x,
    progress = interactive()), id = basename(tempfile()),
    language = "en", description = "Book Files"))
BigramTokenizer <-
  function(x)
    unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)
bigram_textdata <- tau::textcnt(
  if(stop_words==T) { bigram_textdata=gsub("[^[:alnum:] ]", "",bigram_textdata)
    tm::removeWords(tm::scan_tokenizer(bigram_textdata),tm::stopwords("english")
                    )}
  else {
    tm::scan_tokenizer(bigram_textdata)
  }, split = "[[:space:][:punct:][:digit:]]+",method = "string", n = 2L, lower = 1L)
bigram_textdata <- plyr::ldply(bigram_textdata, data.frame)
colnames(bigram_textdata)<-c("word", "frequency")
bigram_textdata$text_percent <- bigram_textdata$frequency / sum(bigram_textdata$frequency)*100
new_bigram_textdata <- bigram_textdata[order(-bigram_textdata$frequency),]
```

According to head of frequency table of word pairs, the most frequency word pair in documents. is "said, mr",whose freuqnecy is 5087 and the $P(mr, said) = 0.387\%$. In order to calculate the bigram model, we divided the joint probability by the probability of unigram.

```
head(new_bigram_textdata)
```

```
##                word frequency text_percent
## 188863      said mr      5089    0.3698283
## 255809          x x      3573    0.2596574
## 101911       i dont      2408    0.1749944
## 102587       i know      2158    0.1568264
## 144745 mr pickwick      2041    0.1483237
## 103742      i think      1931    0.1403298
```

```
bigword<-data.frame(words_bi=new_bigram_textdata$word,new_bigram_textdata$text_percent)

bigword<-bigword %>% separate(words_bi, c("first_word", "word"),sep = "[^[:alnum:]]+")

oneword<-data.frame(new_onegrame_textdata$word,new_onegrame_textdata$text_percent)
colnames(oneword) <- c('word','one_percent')
words_dictionary<-merge(bigword, new_onegrame_textdata,by='word')
words_dictionary$joint<-words_dictionary$new_bigram_textdata.text_percent/words_dictionary$text_percent
words_dictionary<-subset(words_dictionary, select = 1:3)
colnames(words_dictionary)<-c('word','second','conditional_prob')
```

By calculating the bi gram probability, we can know the probability of words we picked given the last word from sequences. For example, the bigram probability of the "daling"" based on the last word is the "a" is $P(W_n = short|W_{n-1} = a) = 0.0001520856\%$, which has very small choice appear in Charles Dickens's novel text files.

```
head(words_dictionary)
```

```
##   word    second conditional_prob
## 1    a     money     0.0001453442
## 2    a    hearth     0.0002180163
## 3    a      bear     0.0001453442
## 4    a        wn     0.0002180163
## 5    a        cx     0.0005087047
## 6    a companion     0.0001453442
```

After the bigram calculation, we selected the initial word as "i", and generated the sentence in the below:

```
generate<-function(init,words_dictionary){
  sentence<- c()
  while(is.na(init)==FALSE){
    sentence<-c(sentence,init)
    secondword<-words_dictionary[words_dictionary$word==init,]
    list_second<- setdiff(secondword[order(secondword$conditional_prob,
                          decreasing = TRUE),]$second,sentence)
    init<-list_second[1]
  }
  return(toupper(paste(unlist(sentence), collapse=' ')))
}

init=new_onegrame_textdata$word[1]
generate(init,words_dictionary)
```

```
## [1] "I SAID SIR YES OH DEAR MY SAY DARE HOW KNOW DONT YOU NOW JUST THATS WELL PRETTY A DCOLOR BBACKG
```

## Conclusion

Language models contribute a process to distribute a probability to a sentence or a seuqnece of words, and to predict a word from leading words. n-grams models are Markov models which predict words from a fixed number of preceding words. In addition,the probabilities in n-gram models could be estimated by counting number of words occurence based on paricular pervious text condition. In this project, we implemented unigram and bigram models,applied both models with text documents of Charles Dickens and generated a new word sequence based on our analysis result.As we can see, both unigram and bigram models provided some particular features of text document written by Charles Dickens. As increase the size of preceding windows, there would be a more specific pattern once we implement models into different data set. This phenomenon indicates one of most common dilemmas in data analysis–there is always compromision between approxmation accuracy and computation complexity.

## GitHub Repository

https://github.com/Adelingtan/stat610-final