# Reproducible Report Submission

*Shane Kao*

*Sunday, February 22, 2015*

## Conditional Probability

I use very simple concept, condition probability, to complete the predictive model. For example, if user input "a lot of", then the probability of the next word is "people" is

$$P(next\ word\ is\ people|a\ lot\ of) = \frac{P(a\ lot\ of\ people)}{P(a\ lot\ of)}$$

The equation gives us 3-gram example, can be apply to $n$-gram, and it can implement easily in R as follows:

```
setwd("C:/Users/asus/Downloads/final/en_US")
news=readLines("news_clean.txt",n=100000)
length(grep("a lot of people",news))/length(grep("a lot of .",news))
```

```
## [1] 0.08508509
```

In general, we want to output the most likely outcome, such as 10 highest probability words

```
x=news[grep("a lot of .",news)]
word=c()
for(i in 1:length(x)){
        word=append(word,strsplit(strsplit(x[i],"a lot of ")[[1]][2]," ")[[1]][1])
}
word<-gsub("\"","",word)
output=as.data.frame(table(word))
output$probability<-output$Freq/length(x)
head(output[order(output$probability,decreasing=TRUE),c("word","probability")],10)
```

```
##          word probability
## 273    people  0.08008008
## 423      time  0.03803804
## 418    things  0.03103103
## 413       the  0.02202202
## 160      good  0.02102102
## 237     money  0.02002002
## 154       fun  0.01601602
## 101 different  0.01201201
## 462      work  0.01201201
## 200        it  0.01101101
```

## Model Fitting

In order to make the model smaller and more efficient, we drop lines doesn't contain the top ten frequencies 3-grams. Randomly assign 60% of data to training set, 20% to test set and 20% to validation set.

## Training Set

This data set is used to fit the original model.

```
pred=function(input,n){
        input<-tolower(input)
        input<-gsub("[^0-9a-z\ ]","",input)
        pattern=paste(paste(rev(rev(strsplit(input," ")[[1]])[1:n]),collapse = " "),".",sep=" ")
        x=train_data[grep(pattern,train_data)]
        if(length(x)!=0){
                word=c()
                for(i in 1:length(x)){
                        seg=paste0(paste(rev(rev(strsplit(input," ")[[1]])[1:n]),collapse = " "),"  ")
                        word=append(word,strsplit(strsplit(x[i],seg)[[1]][2]," ")[[1]][1])
                }
                word<-gsub("\"","",word)
                output=as.data.frame(table(word))
                output$probability<-output$Freq/length(x)
                head(output[order(output$probability,decreasing=TRUE),c("word","probability")],10)
        }else{
                data.frame("word"="","probability"="")
             }
}
```

For example, if user input "I am afraid I won't be able to" and use 3-gram, then the output is

```
pred("I am afraid I won't be able to",3)
```

```
##      word probability
## 437   get  0.05520027
## 294    do  0.04123191
## 578  make  0.03382699
## 861   see  0.02642208
## 445    go  0.02423426
## 398  find  0.01868058
## 1043  use  0.01413665
## 978  take  0.01363177
## 671  play  0.01312689
## 475  help  0.01295860
```

In some cases people type words does not appear in the corpora, then the output is

```
pred("el3vul4",1)
```

```
##   word probability
## 1
```

## Test Set

This data set is used to predict, if the output doesn't contain the next word of top ten frequencies 3-grams, then add the line to training set. For example, "I'm afraid I won't be able to come",and "come" doesn't in the output, then we move this sentece to training set.

```
pattern=c("one of the","a lot of","to be a","i want to","be able to","out of the",
  "going to be","some of the","as well as","the fact that")
for(i in 1:10){
        output=pred(pattern[i],3)$word
        index=grep(paste(pattern[i],".",sep=" "),test_data)
        test_freq_word=test_data[index]
        add_index=c()
        for(j in 1:length(index)){
                word=strsplit(strsplit(test_freq_word[j],paste(pattern[i]," ",sep=""))[[1]][2]," ")[[1]
                word<-gsub("[^0-9a-z\ ]","",word)
                if(!word %in% output){
                        add_index=append(add_index,j)
                }
        }
        train_data=append(train_data,test_freq_word[add_index])
}
```

### Validation Set

This data set is used to check the model performance, notice that if the next word in function `pred` output, then we think the prediction is correct.

```
pattern=c("one of the","a lot of","to be a","i want to","be able to","out of the",
  "going to be","some of the","as well as","the fact that")
n=0
m=c()
for(i in 1:10){
        output=pred(pattern[i],3)$word
        index=grep(paste(pattern[i],".",sep=" "),validation_data)
        val_freq_word=validation_data[index]
        n=n+length(val_freq_word)
        for(j in 1:length(index)){
                word=strsplit(strsplit(val_freq_word[j],paste(pattern[i]," ",sep=""))[[1]][2]," ")[[1]]
                word<-gsub("[^0-9a-z\ ]","",word)
                if(word %in% output){
                        m=append(m,j)
                }
        }
}
length(m)/n
```

```
## [1] 0.3203547
```

## Discussion

We use the function `pred` to build a predictive model, the accuracy is **32%**, it's pretty low, but the model just gives ten possible outcome, we can expect higher accuracy if the model give us more outcome. The method I use is strongly rely on data, it is unable to handle cases where a particular $n$-gram isn't observed, but the adventage is simple and intuitive.