TeamD\_Case4

Shine, Jason Meng, Tianyu Zhang, Lingyi Kong

4/3/2018

## Load the packages

## Loading required package: NLP

## ── Attaching packages ────────────────────────────────────────────────────────────────────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 2.2.1 ✔ purrr 0.2.4  
## ✔ tibble 1.4.2 ✔ dplyr 0.7.4  
## ✔ tidyr 0.8.0 ✔ stringr 1.3.0  
## ✔ readr 1.1.1 ✔ forcats 0.3.0

## ── Conflicts ───────────────────────────────────────────────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ ggplot2::annotate() masks NLP::annotate()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

##   
## Attaching package: 'qdapRegex'

## The following object is masked from 'package:dplyr':  
##   
## explain

## The following object is masked from 'package:ggplot2':  
##   
## %+%

## Loading required package: qdapTools

##   
## Attaching package: 'qdapTools'

## The following object is masked from 'package:dplyr':  
##   
## id

## Loading required package: RColorBrewer

##   
## Attaching package: 'qdap'

## The following object is masked from 'package:forcats':  
##   
## %>%

## The following object is masked from 'package:stringr':  
##   
## %>%

## The following object is masked from 'package:dplyr':  
##   
## %>%

## The following object is masked from 'package:purrr':  
##   
## %>%

## The following object is masked from 'package:tidyr':  
##   
## %>%

## The following objects are masked from 'package:tm':  
##   
## as.DocumentTermMatrix, as.TermDocumentMatrix

## The following object is masked from 'package:NLP':  
##   
## ngrams

## The following object is masked from 'package:base':  
##   
## Filter

## Loading required package: grid

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:qdap':  
##   
## %>%

## The following object is masked from 'package:purrr':  
##   
## set\_names

## The following object is masked from 'package:tidyr':  
##   
## extract

##   
## Attaching package: 'igraph'

## The following objects are masked from 'package:qdap':  
##   
## %>%, diversity

## The following objects are masked from 'package:dplyr':  
##   
## as\_data\_frame, groups, union

## The following objects are masked from 'package:purrr':  
##   
## compose, simplify

## The following object is masked from 'package:tidyr':  
##   
## crossing

## The following object is masked from 'package:tibble':  
##   
## as\_data\_frame

## The following objects are masked from 'package:stats':  
##   
## decompose, spectrum

## The following object is masked from 'package:base':  
##   
## union

## Data Prep

## 'data.frame': 120787 obs. of 6 variables:  
## $ listing\_id : int 7441144 7441144 7441144 7441144 7441144 12233830 12233830 12233830 12233830 12233830 ...  
## $ id : int 42015616 52339991 53933755 79106284 81799307 70018232 70649304 70882891 71725720 74530572 ...  
## $ date : chr "8/10/15" "10/28/15" "11/14/15" "6/11/16" ...  
## $ reviewer\_id : int 38507747 45395024 36548987 39468163 39468163 2134782 37567875 68058568 43639502 58310922 ...  
## $ reviewer\_name: chr "Jermaine" "Donato" "Jason" "Michael" ...  
## $ comments : chr "Pretty nice, quiet, cozy place to stay. Toiletries, snacks, coffee, WiFi, cable TV, iron was all included. One "| \_\_truncated\_\_ "The host was extremely welcoming and obliging. The neighborhood is quiet and charming, perfect for a quiet visi"| \_\_truncated\_\_ "Nice and easy stay - with good accommodations especially the cable TV " "The host has been very accommodating and helpful. The description in the ad is accurate. The room is very clean"| \_\_truncated\_\_ ...

## listing\_id id date reviewer\_id reviewer\_name   
## 0 0 0 0 0   
## comments   
## 0

### Interpration:

Our first step is to load the Airbub dataset and set stringAsFactors as False. This can efficiently prevents conversion of string to factors and treats them as vectors. Then we check the dataset by scanning its structure(str function). Missing data can be a serious problem as we process data, so our next step is to check total missing data. Fortunately, we found that there is no missing data in this reviews file. Following the instruction, we then select a random subset of 1,200 reviews from the dataset to facilicate our further analysis.

## Data Analysis

## all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity  
## 1 all 1200 63744 0.871 0.535 1.627

## [1] 4763 2

## Docs  
## Terms character(0) character(0)  
## “entire 3.371544e-05 0.0000000000  
## abbiamo 3.371544e-05 0.0000000000  
## abends 0.000000e+00 0.0009775171

### Interpretation:

Before our major analysis, we first take a look at the overall polarity of the 1200 random samples. As is showed in the results, the average polarity is 0.871, indicating that the samples are quite positive overall. Then we assign the polarity of each comment to a new column “polarity” and according to the polarity values, we use filter to separate comments into positive comments (polarity > 0) and negative comments (polarity < 0). After transforming the dataframes of comments into terms, we combine the terms and then transform into corpus. When converting the corpus into a term document matrix(tdm), we remove punctuations, numbers and our customized stopwords, and use TfIdf weighting. Then the tdm is converted into a matrix for the following analysis. The dimension and the first three rows are showed in the result table.

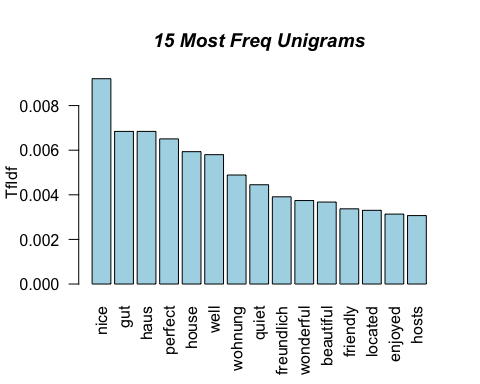
### Discussion about mystopwords:

* Stopwords: \*Dimension: #row(5008), #col(2)

## Visualize 15 most frequent single terms

## “entire abbiamo abends   
## 3.371544e-05 3.371544e-05 9.775171e-04

## nice gut haus perfect house well   
## 0.009204316 0.006842620 0.006842620 0.006507080 0.005933918 0.005799056   
## wohnung quiet freundlich wonderful beautiful friendly   
## 0.004887586 0.004450438 0.003910068 0.003742414 0.003674983 0.003371544   
## located enjoyed hosts   
## 0.003304113 0.003135536 0.003068105

 ### Interpration: In order to visualize the 15 most frequent single terms, first we calculate the row sums of the matrix we just created to get the frequency of each term. Then we sort the term frequency in descending order to get a clearer picture for the most common words. We viewed the top 15 most frequent words and created a barchart based on them. Because several words such as “really”, “highly”, or “definitely” have high frequency of occurrence but low enlightening meaning, so our group delete these “meaningless” words by adding them to stopwords function. After erasing these disturbing words, we generate a clear and meaningful barplot for the 15 most frequent single words. From the output we can see that the results are both in English and in German.Gut refers to good in English;haus refers to house in English;wohnug refers to apartment in English;freundlich refers to friendly in English.

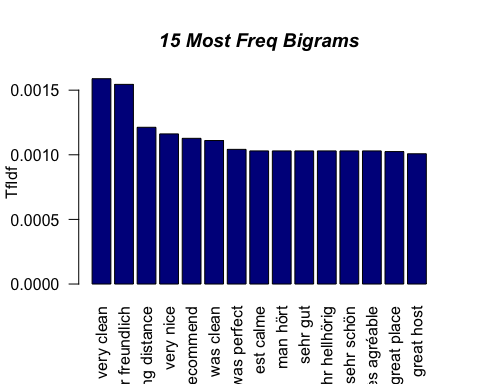
## Visualize 15 most frequent bigrams

## [1] 27832 2

## Docs  
## Terms 1 2  
## accurate 1.708117e-05 0  
## adults 5.124351e-05 0  
## advice 1.708117e-05 0

## accurate adults advice   
## 1.708117e-05 5.124351e-05 1.708117e-05

## very clean sehr freundlich walking distance very nice   
## 0.001588549 0.001544799 0.001212763 0.001161520   
## highly recommend was clean was perfect est calme   
## 0.001127357 0.001110276 0.001041951 0.001029866   
## man hört sehr gut sehr hellhörig sehr schön   
## 0.001029866 0.001029866 0.001029866 0.001029866   
## très agréable great place great host   
## 0.001029866 0.001024870 0.001007789

 ### Interpretation: To plot a barplot of the most frequent bigrams, we set the term length to 2, create a new term document matrix, and convert it to a new matrix. Same as the part of unigrams, we use TfIdf weighting, and remove punctuations, numbers and our customized stopwords. In our first attempt with only one-word stopwords, we find that the stopwords don’t work for bigrams and the regular stopwords ,like “der”(“the” in Germany), that are deleted in the single-word analysis show up again in the bigram analysis. Most frequent bigrams are like “and I”, “der haus”(the house in Germany), “it is” that make little sense. Therefore we add two-word stopwords to our customized stopwords, “mystopwords”, so that every bigram in the bar plot provides some information. Then we show the 15 most frequent bigrams and draw a bar chart to show the phrases and their frequencies.

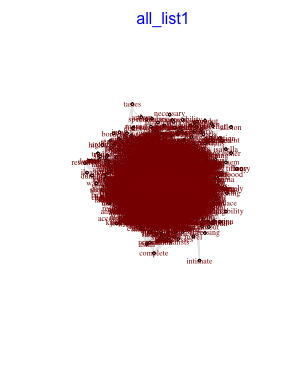
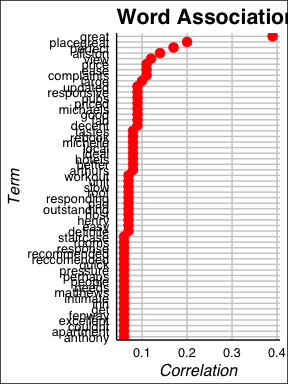
### Comments on the short phrases

The 15 most frequent meaningful phrases contain 7 English phrases, 7 German phrases and 1 French phrases. Their meanings are “very clean”, “very friendly”, “walking distance”, “very nice”, “highly recommend”, “was clean”, “was perfect”, “is calm”, “very good”, “very light-colored”, “very beautiful”, “very nice”, “great place”, “great host”, and “perfect for”. It’s obvious that all of the most frequent phrases are positive, indicating that overall Airbnb is a good source of temporary accomodation. When we dig further, we find that the aspects people care most are whether the house/apartment is clean, whether the host is friendly, and how convinient is the house/apartment located. The languages of the frequent phrases can also convey some information. The high frequency of German phrases may indicate that either Airbnb is commonly used by German speakers or they are more willingly to leave a comment.

### Comparison to the single-word analysis

## Create a new Corpus

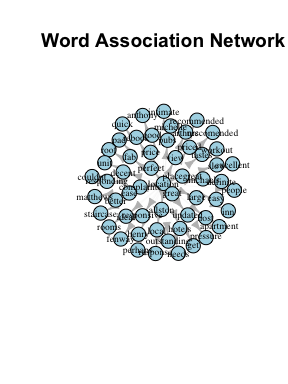
## Visualize word association network for “location”



##   
## Match Terms  
## ===========

##   
## List 1:  
## location, locations, neighborhoodlocation, locationclose, locationvery, locationwise

##

 ### Interpretation: Firstly, we use the findAssocs() function to take a look at the words that are related with “location”. We set the correlation to 0.55 to filter the words, so the size of the target word group does not get too big. Then we tried to use the “word\_associate” function to create a word netword plot, which appears to be overcomplicated and unrecognizable. Thus, we decided to simplify the plot using basic “plot” function. In this plot, correlation is used to limit the number of words, and the relationship between these major words and “location” is clear. The distance represents the ###### , and the width of the string represents the degree of correlation.

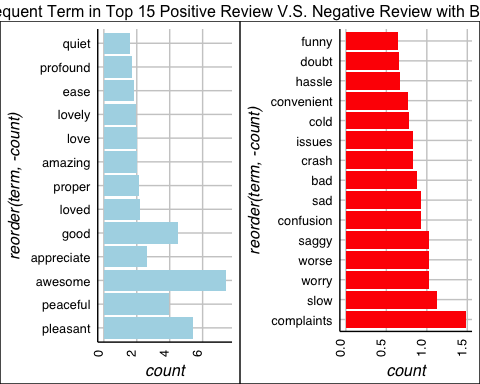
### Comments on the word association plots:

The reason we decided to create a simplier plot is that we found the nature of the “word\_association” function made the network map complicated. We originally planned to use correlation to limit the number of words in the net work map, but again, the “word\_association” function did not allow us to do so, since it is designed to take every word from the comments that contains string into the map. The stopwords we have helped to decrease some meaningless words, but the word base is still too big to be presented clearly. Thus, we concluded that the network map has to be complicated in this case, as the nature of the “word\_association” function drives.

## Pyramid plot by using Bing

## # A tibble: 1 x 3  
## document term count  
## <chr> <chr> <dbl>  
## 1 1 非常熱心的房東 5.11

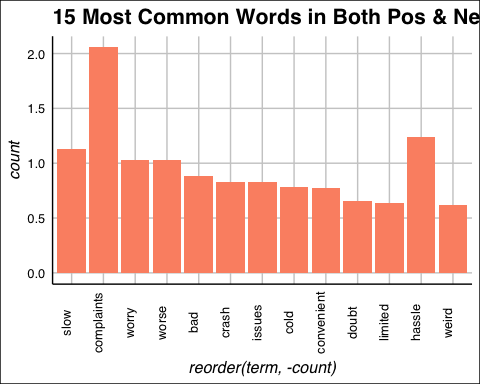
## Joining, by = "document"  
## Joining, by = "document"

 ## Plot for common words in both positive and negative comments by using Bing

bing\_com\_word <- bing\_neg\_term[bing\_neg\_term$term %in% bing\_pos\_term$term,]%>%  
 arrange(desc(count))  
bing\_com\_plot <- ggplot(  
 bing\_com\_word[1:15,], aes(x=reorder(term,-count), count)) +  
 geom\_bar(stat = "identity", fill = "#fc9272") +   
 theme\_gdocs() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = -0.1)) +  
 ggtitle("15 Most Common Words in Both Pos & Neg Reviews(Bing)")  
 coord\_flip()

## <ggproto object: Class CoordFlip, CoordCartesian, Coord>  
## aspect: function  
## distance: function  
## expand: TRUE  
## is\_linear: function  
## labels: function  
## limits: list  
## range: function  
## render\_axis\_h: function  
## render\_axis\_v: function  
## render\_bg: function  
## render\_fg: function  
## train: function  
## transform: function  
## super: <ggproto object: Class CoordFlip, CoordCartesian, Coord>

bing\_com\_plot



## Pyramid plot by using AFINN

afinn\_lex <- get\_sentiments("afinn")  
head(afinn\_lex,3)

## # A tibble: 3 x 2  
## word score  
## <chr> <int>  
## 1 abandon -2  
## 2 abandoned -2  
## 3 abandons -2

# Join text to lexicon  
all\_terms\_afinn\_words <- inner\_join(all\_tidy, afinn\_lex, by = c("term" = "word"))  
  
head(all\_terms\_afinn\_words,3)

## # A tibble: 3 x 4  
## document term count score  
## <chr> <chr> <dbl> <int>  
## 1 2 beautiful 0.120 3  
## 2 2 clean 0.0580 2  
## 3 2 drop 0.235 -1

#aggregate  
all\_tidy\_sentiment2\_agg <- all\_terms\_afinn\_words %>%   
 # Group by document  
 group\_by(document) %>%  
 # Sum scores by document  
 summarize(total\_score = sum(score)) %>%  
 ungroup()  
  
head(all\_tidy\_sentiment2\_agg,3)

## # A tibble: 3 x 2  
## document total\_score  
## <chr> <int>  
## 1 10 13  
## 2 100 17  
## 3 1000 9

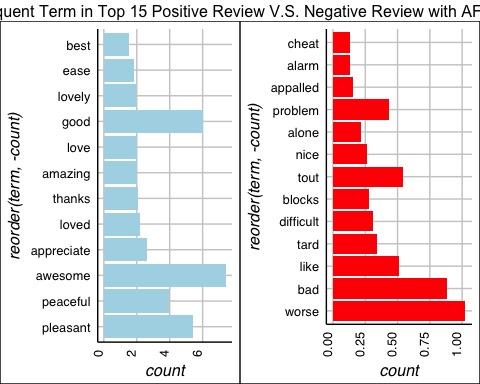
# Add polarity  
all\_tidy\_pol2 <- all\_tidy\_sentiment2\_agg %>%   
 mutate(  
 pol = ifelse(total\_score>0, "positive", "negative")  
 )  
  
#subset top 1positive comments  
afinn\_pos <- all\_tidy\_pol2[all\_tidy\_pol2$pol=="positive",]   
  
afinn\_pos\_term <- inner\_join(afinn\_pos, all\_terms\_afinn\_words) %>%  
 arrange(desc(count))

## Joining, by = "document"

#subset negative comments  
afinn\_neg <- all\_tidy\_pol2[all\_tidy\_pol2$pol=="negative",]  
  
afinn\_neg\_term <- inner\_join(afinn\_neg, all\_terms\_afinn\_words) %>%  
 arrange(desc(count))

## Joining, by = "document"

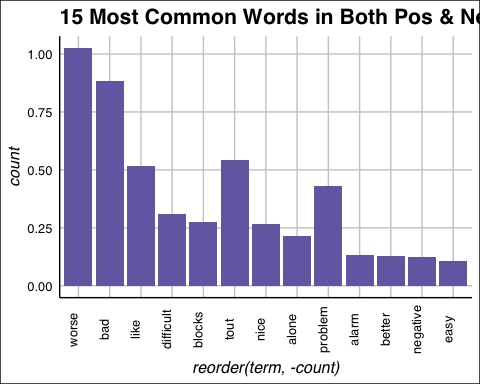
#plot top 15 positive and negative terms  
afinn\_pos\_plot <- ggplot(  
 afinn\_pos\_term[1:15,], aes(x=reorder(term,-count), count)) +  
 geom\_bar(stat = "identity", fill = "light blue") +   
 theme\_gdocs() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = -0.1)) +   
 coord\_flip()  
  
afinn\_neg\_plot <- ggplot(  
 afinn\_neg\_term[1:15,], aes(x=reorder(term,-count), count)) +  
 geom\_bar(stat = "identity", fill = "red") +   
 theme\_gdocs() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = -0.1)) +  
 coord\_flip()  
  
grid.arrange(afinn\_pos\_plot, afinn\_neg\_plot, nrow = 1, top = "Frequent Term in Top 15 Positive Review V.S. Negative Review with AFINN")

 ## Plot for common words in both positive and negative comments by using AFINN

com\_word\_afinn <- afinn\_neg\_term[afinn\_neg\_term$term %in% afinn\_pos\_term$term,]%>%  
 arrange(desc(count))  
afinn\_com\_plot <- ggplot(  
 com\_word\_afinn[1:15,], aes(x=reorder(term,-count), count)) +  
 geom\_bar(stat = "identity", fill = "#756bb1") +   
 theme\_gdocs() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = -0.1)) +  
 ggtitle("15 Most Common Words in Both Pos & Neg Reviews(AFINN)")  
 coord\_flip()

## <ggproto object: Class CoordFlip, CoordCartesian, Coord>  
## aspect: function  
## distance: function  
## expand: TRUE  
## is\_linear: function  
## labels: function  
## limits: list  
## range: function  
## render\_axis\_h: function  
## render\_axis\_v: function  
## render\_bg: function  
## render\_fg: function  
## train: function  
## transform: function  
## super: <ggproto object: Class CoordFlip, CoordCartesian, Coord>

afinn\_com\_plot



## Word Cloud by Bing

bing\_term\_pos <- paste(bing\_pos\_term$term, collapse = " ")  
bing\_term\_neg <- paste(bing\_neg\_term$term, collapse = " ")  
  
term\_m <-matrix(c(bing\_term\_pos,bing\_term\_neg),1,2)  
a\_corpus <- VCorpus(VectorSource(term\_m))  
  
a\_tdm <- TermDocumentMatrix(a\_corpus)  
  
a\_tdm\_m <- as.matrix(a\_tdm)  
  
colnames(a\_tdm\_m) <- c("positive", "negative")  
  
comparison.cloud(  
 a\_tdm\_m,   
 max.words = 100,  
 colors = c("darkblue", "darkred")  
)

