A3: Business Insight Report

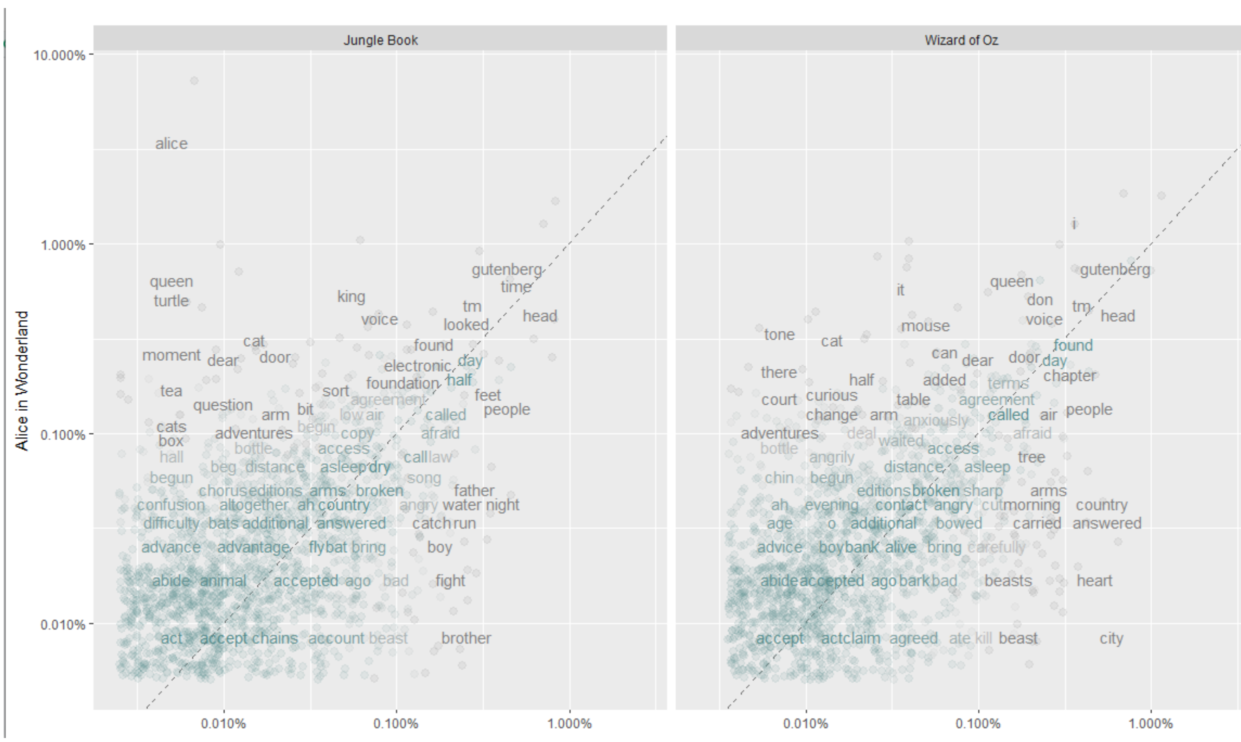
Text Analytics and Natural Language Processing (NLP)

Prof. Thomas Kurnicki

Christopher Pramodh

The market for children's toys consists of 60 million children with an average of 70 toys per child. Parents spend an average 6500$ on toys until their child turns 18. Children play longer with higher resolution toys and enhanced technology is now expected in toys. With toys containing Bluetooth speakers, GPS tracking and mobile device integration, companies are developing more innovative toys every day. Lego provides audio instructions for its building blocks and even augmented reality board games are coming to the market. (Team Linchpin, 2020) Inspired by children's literature classics like Alice in Wonderland, The Jungle Book and The Wizard of Oz, XYZ Toys Manufacturing Ltd wants to develop an original toy which interacts creatively through speech with the player. We shall analyze the words used frequently in these classics and analyze the sentiment that they exhibit.

Correlogram comparing frequent words used in all three texts



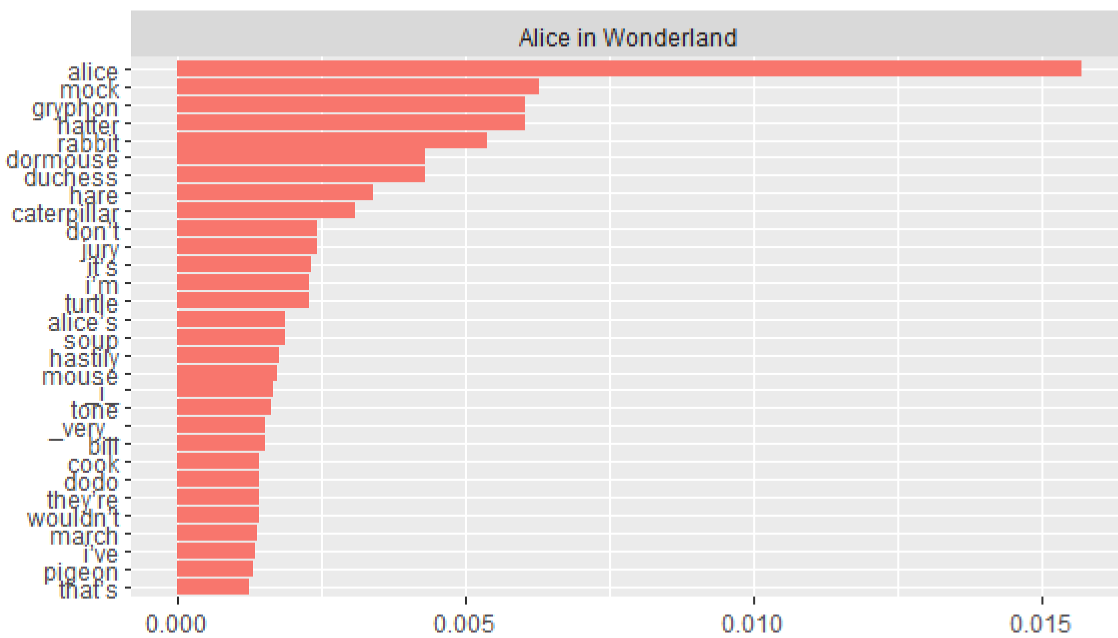
The highly frequent words between Alice in Wonderland and Jungle book are 'time', 'head', 'looked' and 'day'. Jungle book uses words such as 'brother', 'fight', 'boy', which Alice in Wonderland does not. This is probably because protagonists of both novels are of different gender. Jungle book tells the story of brotherhood where Mowgli, a human, leaves his family, a pack of wolves, due to threat from a tiger called Sher Khan.

The highly frequent words between Alice in Wonderland and Wizard of Oz are ‘voice’, ‘head’, ‘found’ and ‘day’. Wizard of Oz uses words such as ‘heart’, ‘city’, ‘beast’ while Alice in Wonderland does not.

Overall, visually the Wizard of Oz plot is slightly denser than Jungle Book plot. In fact, the correlation in words between Alice in Wonderland and Wizard of Oz is 0.532 which is greater than the correlation between Alice in Wonderland and Jungle book which is 0.315. This could be because both Alice in Wonderland and Wizard of Oz have female protagonists Alice and Dorothy. The highly frequent words are similar in both plots. We can see that the three texts are not highly correlated which could be due to the difference in themes, style, and tone, even though they all fall under children’s fiction.

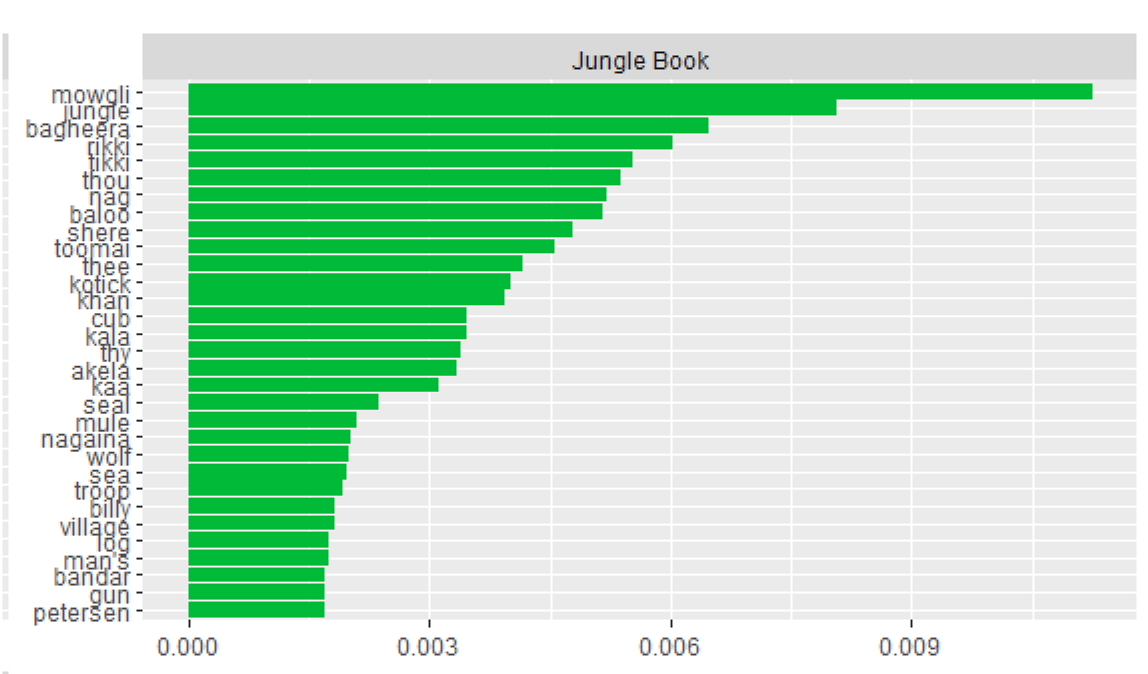
Using the TF-IDF technique where we try to determine importance from less frequent words, let’s try to analyze the most important words in the books.

TF-IDF plot for Alice in Wonderland



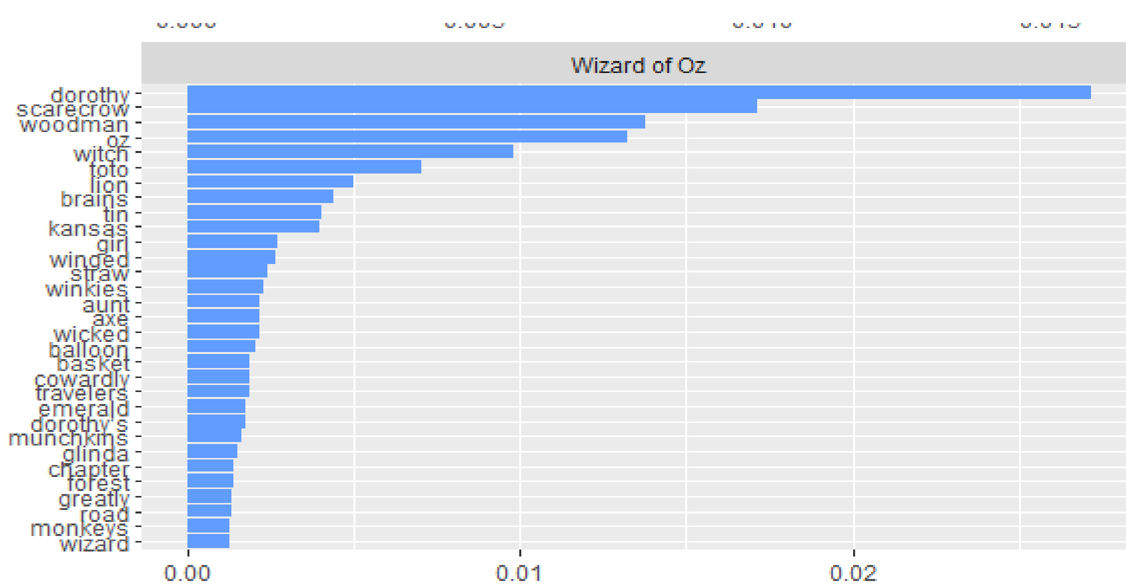
For Alice in Wonderland the top five words are ‘alice’, ‘mock’, ‘gryphon’, ‘hatter’ and ‘rabbit’. Alice, Hatter and Rabbit are primary characters in the novel.

TF-IDF plot for Jungle Book



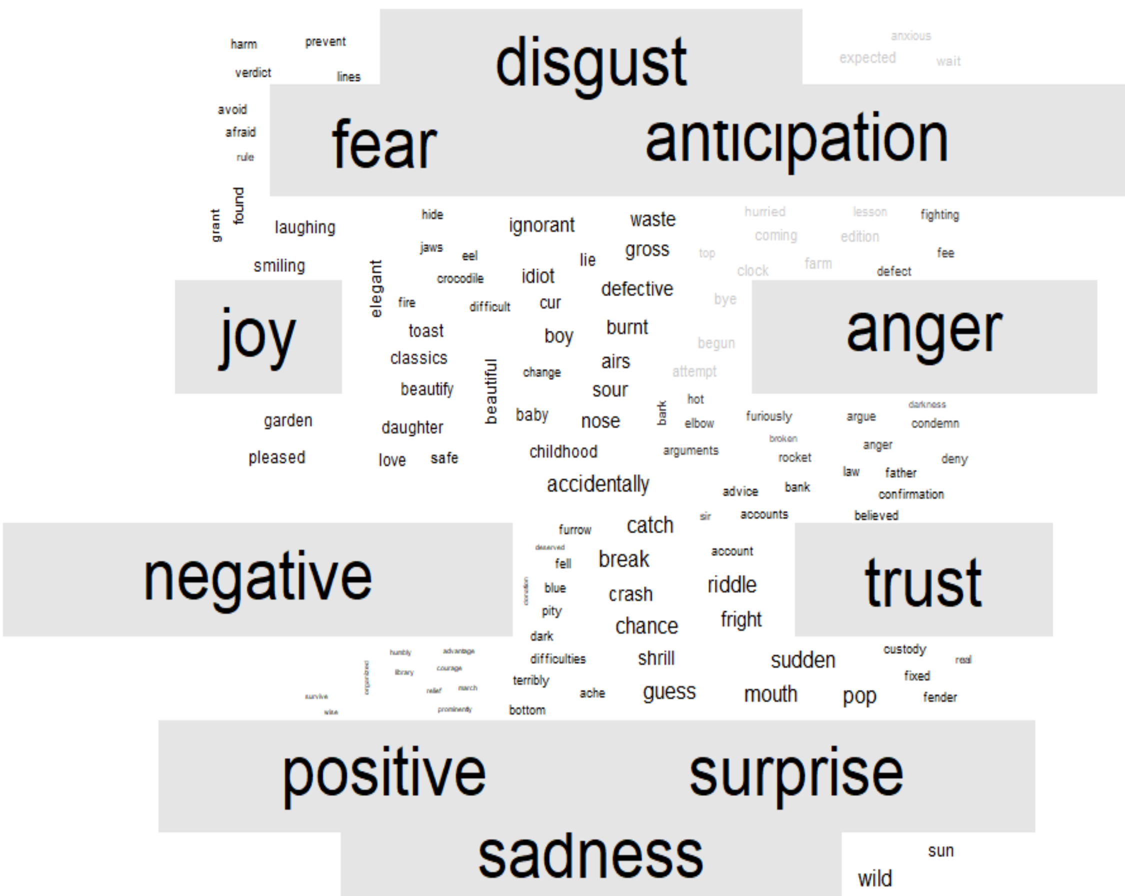
For Jungle Book, the top five words are ‘mowgli’, ‘jungle’, ‘bagheera’, ‘rikki’ and ‘tikki’. Mowgli, Bagheera, Rikki and Tikki are primary characters in Jungle Book.

TF-IDF plot for Wizard of Oz

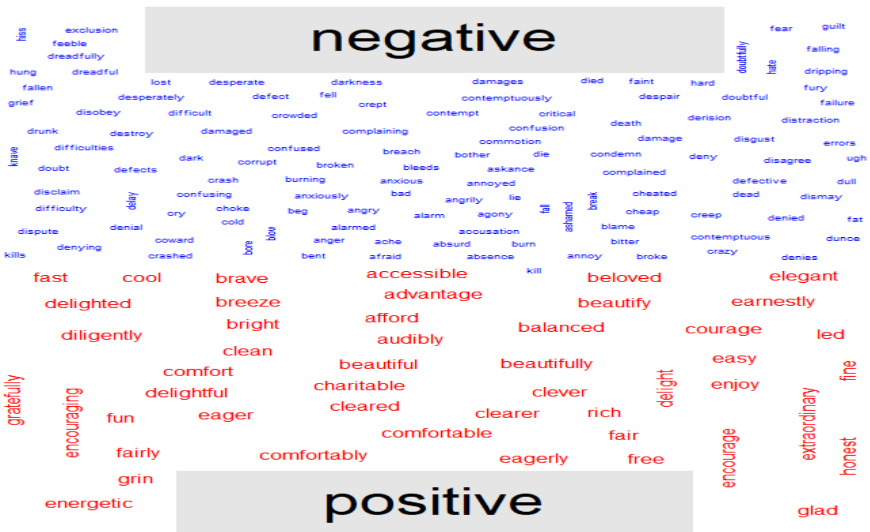


For Wizard of Oz the top five words are ‘dorothy’, ’scarecrow’, ’woodman’, ’oz’ and ‘witch’. These five words are primary characters in Wizard of Oz. Overall, the important character names are shown as most important to the books. Repetition of character names builds a sense of affinity and makes readers care about the characters.

Now let us try to understand the emotions that these classics try to exhibit. From our word cloud below we can see that the larger font words are mostly siding with the emotions of ‘surprise’, ‘trust’, ‘disgust’ and anticipation. Despite being children’s literature, these novels not only focus on positive emotions such as joy and trust but also negative emotions such as disgust and anticipation to further enhance the story-telling.



Furthermore, lets bi-furcate the words into positive and negative sentiments. Once again, we can see that there are a lot more negative words compared to positive which could be a technique to enhance the story with real stakes.



Using our TF-IDF technique let's try to analyze two-word combinations or bigrams which are most important to the texts. We can see that they are character references where the first word is descriptive while the second word is the character, for example, white rabbit or kala nag.

	book	bigram	n	tf	idf	tf_idf
1	Wizard of Oz	tin woodman	107	0.037425673	1.098612	0.041116305
2	Alice in Wonderland	mock turtle	50	0.023496241	1.098612	0.025813259
3	Wizard of Oz	wicked witch	56	0.019587268	1.098612	0.021518814
4	Wizard of Oz	emerald city	53	0.018537950	1.098612	0.020366020
5	Jungle Book	rikki tikki	94	0.018139714	1.098612	0.019928513
6	Alice in Wonderland	march hare	31	0.014567669	1.098612	0.016004220
7	Jungle Book	shere khan	71	0.013701274	1.098612	0.015052388
8	Wizard of Oz	winged monkeys	28	0.009793634	1.098612	0.010759407
9	Alice in Wonderland	white rabbit	20	0.009398496	1.098612	0.010325303
10	Jungle Book	kala nag	47	0.009069857	1.098612	0.009964257

When using three-word combinations or trigrams we see that Alice in Wonderland dominates the list where many of the combinations are about how the protagonist reacts in situations.

	book	trigram	n	tf	idf	tf_idf
1	Alice in Wonderland	beau ootiful soo	4	0.008695652	1.0986123	0.009553150
2	Alice in Wonderland	ootiful soo oop	4	0.008695652	1.0986123	0.009553150
3	Alice in Wonderland	white kid gloves	4	0.008695652	1.0986123	0.009553150
4	Wizard of Oz	winged monkeys flew	4	0.007476636	1.0986123	0.008213924
5	Alice in Wonderland	cats eat bats	3	0.006521739	1.0986123	0.007164863
6	Alice in Wonderland	alice cautiously replied	2	0.004347826	1.0986123	0.004776575
7	Alice in Wonderland	alice hastily replied	2	0.004347826	1.0986123	0.004776575
8	Alice in Wonderland	alice i'm glad	2	0.004347826	1.0986123	0.004776575
9	Alice in Wonderland	beautiful beautiful soup	2	0.004347826	1.0986123	0.004776575
10	Alice in Wonderland	dear cried alice	2	0.004347826	1.0986123	0.004776575

From our analysis we have found that there is no strong correlation between the texts. This is because the stories, themes and styles are handled differently and uniquely in each case. We also see that there is frequent repetition of prominent character names throughout the texts. We found that the books exhibited emotions of 'surprise' and 'anticipation' which are crucial for story progression. We also found that there were a lot of negative words used in the texts contributing to the narrative of the story. The poetic nature of the bigrams and the trigrams lend a helping hand to the readers' imagination with the use of literary techniques such as alliteration and meter. These aspects have made these classics beloved among children.

The insight we get from this analysis is that toys must have a strong story component. This is probably why movie-based action figures have grown about 20% in 2019. (Team Linchpin, 2020) Since XYZ Toy Manufacturing company are developing an original toy, they would have to develop a strong story component. Since the toy will be using speech, it can be a storyteller using descriptive and poetic language which can capture the imagination of children. The story must have real stakes so that it builds anticipation among children and keeps them engaged.

Bibliography

Team Linchpin. (2020, October 12). *Trends Shaping The Toy Industry Outlook In 2021*. Retrieved from https://linchpinseo.com/: https://linchpinseo.com/trends-shaping-the-toy-industry/

Code from R studio

```
#######################################################################
################   Importing Libraries #############################
#######################################################################
library(wordcloud)
library(dplyr)
library(tidyverse)
library(tidytext)
library(stringr)
library(tidyr)
library(ggplot2)
library(scales)
library(textdata)
library(gutenbergr)
library(reshape2)
#######################################################################
################ Downloading books##### #############################
#######################################################################
alice<- gutenberg_works(title == "Alice's Adventures in Wonderland") %>%
  gutenberg_download(strip = FALSE,mirror="http://mirrors.xmission.com/gutenberg/")
oz<- gutenberg_works(title == "The Wonderful Wizard of Oz") %>%
  gutenberg_download(strip = FALSE,mirror="http://mirrors.xmission.com/gutenberg/")
```

```
jungle<- gutenberg_works(title == "The Jungle Book") %>%
  gutenberg_download(strip = FALSE,mirror="http://mirrors.xmission.com/gutenberg/")
#####################################################################
############### Tidying books #####################################
#####################################################################
data(stop_words)
tidy_alice <- alice %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)
tidy_oz <- oz %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)
tidy_jungle <- jungle %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)

#####################################################################
############### Combining all books to find frequency############
#####################################################################
frequency <- bind_rows(mutate(tidy_alice, book="Alice in Wonderland"),
                       mutate(tidy_oz, book= "Wizard of Oz"),
                       mutate(tidy_jungle, book="Jungle Book")
)%>%#closing bind_rows
```

```
  mutate(word=str_extract(word, "[a-z']+")) %>%
  count(book, word) %>%
  group_by(book)%>%
  mutate(proportion = n/sum(n))%>%
  select(-n)  %>%
  spread(book, proportion) %>%
  gather(book, proportion, "Wizard of Oz", "Jungle Book")
####################################################################
############### Plotting Correlogram ############################
####################################################################
ggplot(frequency, aes(x=proportion, y=`Alice in Wonderland`,
                      color = abs(`Alice in Wonderland`- proportion)))+
  geom_abline(color="grey40", lty=2)+
  geom_jitter(alpha=.1, size=2.5, width=0.3, height=0.3)+
  geom_text(aes(label=word), check_overlap = TRUE, vjust=1.5) +
  scale_x_log10(labels = percent_format())+
  scale_y_log10(labels= percent_format())+
  scale_color_gradient(limits = c(0,0.001), low = "darkslategray4", high = "gray75")+
  facet_wrap(~book, ncol=2)+
  theme(legend.position = "none")+
  labs(y= "Alice in Wonderland", x=NULL)
```

```
############################################
##doing the cor.test() ####################
############################################
cor.test(data=frequency[frequency$book == "Jungle Book",],
        ~proportion + `Alice in Wonderland`)
cor.test(data=frequency[frequency$book == "Wizard of Oz",],
        ~proportion + `Alice in Wonderland`)
#######################################################################
### Identifying common word counts among texts#####################
#######################################################################
original_books_t <- bind_rows(mutate(tidy_alice, book="Alice in Wonderland"),
                    mutate(tidy_oz, book= "Wizard of Oz"),
                    mutate(tidy_jungle, book="Jungle Book")) %>%
  count(book, word, sort=TRUE) %>%
  ungroup()
total_words_t <- original_books_t %>%
  group_by(book) %>%
  summarize(total= sum(n))

book_words_t <- left_join(original_books_t, total_words_t)

print(book_words_t)
```

```
#######################################
########## ZIPF's law ################
#######################################
freq_by_rank_t <- book_words_t %>%
  group_by(book) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total)
freq_by_rank_t

#let's plot ZIPF's Law
freq_by_rank_t %>%
  ggplot(aes(rank, `term frequency`, color=book))+
  #let's add a tangent line , the first derivative, and see what the slop is
  geom_abline(intercept=-0.62, slope= -1.1, color='gray50', linetype=2)+
  geom_line(size= 1.1, alpha = 0.8, show.legend = FALSE)+
  scale_x_log10()+
  scale_y_log10()
#####################################################
################# TF_IDF #########################
#####################################################
book_words_t <- book_words_t %>%
  bind_tf_idf(word, book, n)
```

```
book_words_t # we get all the zeors because we are looking at stop words ... too common

book_words_t %>%
  arrange(desc(tf_idf))

#what can we say about these words?

# looking at the graphical apprach:

book_words_t %>%
  arrange(desc(tf_idf)) %>%
  mutate(word=factor(word, levels=rev(unique(word)))) %>%
  group_by(book) %>%
  top_n(30) %>%
  ungroup %>%
  ggplot(aes(word, tf_idf, fill=book))+
  geom_col(show.legend=FALSE)+
  labs(x=NULL, y="tf-idf")+
  facet_wrap(~book, ncol=2, scales="free")+
  coord_flip()
###########################################
##### Creating a word cloud for texts######
##########################################


original_books_tn <- bind_rows(mutate(tidy_alice, book="Alice in Wonderland"),
                               mutate(tidy_oz, book= "Wizard of Oz"),
                               mutate(tidy_jungle, book="Jungle Book")) %>%
```

```
  group_by(book) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(word, regex("^chapter [\\divxlc]",
                                                  ignore_case = TRUE))))%>%
  ungroup() %>%
  filter(book == 'Alice in Wonderland') %>%
  count(word, sort=T)
original_books_tn %>%
  with(wordcloud(word, n, max.words = 100))
#####################################################
#### Adding positive and negative sentiments ######
#####################################################
original_books_tn %>%
  inner_join(get_sentiments('nrc')) %>%
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var='n', fill=0) %>%
  comparison.cloud(colors = c('grey20', 'grey80'),
                   max.words=500,scale=c(1,0.1))
original_books_tn %>%
  inner_join(get_sentiments('bing')) %>%
  count(word, sentiment, sort=TRUE) %>%
  acast(word ~sentiment, value.var='n', fill=0) %>%
  comparison.cloud(colors = c("blue", "red"),
```

```
              max.words=500,scale=c(1,0.1))


######################################################################
################# Creation of bigrams ################################
######################################################################
toy_bigrams <- bind_rows(mutate(alice, book="Alice in Wonderland"),
                         mutate(oz, book= "Wizard of Oz"),
                         mutate(jungle, book="Jungle Book")) %>%
  unnest_tokens(bigram, text, token = 'ngrams', n=2)
toy_bigrams_separated <- toy_bigrams %>%
  separate(bigram, c('word1', 'word2'), sep = " ")
toy_bigrams_filtered <- toy_bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word1 == 'project')%>%
  filter(!word1 == 'gutenberg')%>%
  filter(!word1 == 'tm')%>%
  filter(!word1 == 'archive')%>%
  filter(!word1 == 'literary')%>%
  filter(!word1 == 'NA' | !word2 == 'NA')
toy_bigram_counts <- toy_bigrams_filtered %>%
  count(word1, word2, sort = TRUE)
#want to see the new bigrams
```

```
toy_bigram_counts
###########################################################################
################# Creation of trigrams ################################
###########################################################################
toy_trigram <- bind_rows(mutate(alice, book="Alice in Wonderland"),
                        mutate(oz, book= "Wizard of Oz"),
                        mutate(jungle, book="Jungle Book")) %>%
  unnest_tokens(trigram, text, token = "ngrams", n=3) %>%
  separate(trigram, c('word1', "word2", 'word3'), sep=" ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word3 %in% stop_words$word)%>%
  filter(!word1 == 'project')%>%
  filter(!word1 == 'gutenberg')%>%
  filter(!word1 == 'tm')%>%
  filter(!word1 == 'archive')%>%
  filter(!word1 == 'literary')%>%
  filter(!word1 == 'title')%>%
  filter(!word1 == 'author')%>%
  filter(!word1 == 'release')%>%
  filter(!word1 == 'date')%>%
  filter(!word1 == 'june')%>%
  filter(!word1 == '25')%>%
```

```
  filter(!word1 == '2008')%>%
  filter(!word1 == 'recently')%>%
  filter(!word1 == 'updated')%>%
  filter(!word1 == 'october')%>%
  filter(!word1 == 'character')%>%
  filter(!word1 == 'set')%>%
  filter(!word1 == 'encoding')%>%
  filter(!word1 == 'millennium')%>%
  filter(!word1 == 'fulcrum')%>%
  filter(!word1 == 'ebook')%>%
  filter(!word1 == 'NA' | !word2 == 'NA')
  toy_trigram
#############################################################
###### We can also apply the tf_idf framework  ############
########### on our bigram and trigram #################
#############################################################
# TF-IDF for bigrams
toy_bigram_united <- toy_bigrams_filtered %>%
  unite(bigram, word1, word2, sep=" ") #we need to unite what we split in the previous section
toy_bigram_tf_idf <- toy_bigram_united %>%
  count(book, bigram) %>%
  bind_tf_idf(bigram, book, n) %>%
  arrange(desc(tf_idf))
```

```
toy_bigram_tf_idf

# TF-IDF for trigrams

toy_trigram_united <- toy_trigram %>%

  unite(trigram, word1, word2, word3, sep=" ") #we need to unite what we split in the previous section

toy_trigram_tf_idf <- toy_trigram_united %>%

  count(book, trigram) %>%

  bind_tf_idf(trigram, book, n) %>%

  arrange(desc(tf_idf))

toy_trigram_tf_idf
```