Text Mining

Text Data

Text data is one of the most commonly used types of unstructured data.

Text data is typically generated by users themselves.

Online reviews, movie critics, Tweets, SMS, WhatsApp and WeChat messages, Facebook messages…

Text Data

Yet text data cannot be easily analyzed. For example, how to run a regression with consumer reviews? --- Could this be treated as fixed effects?

Text Data

Yet text data cannot be easily analyzed. For example, how to run a regression with consumer reviews? --- Could this be treated as fixed effects?

We need to extract meaningful measures from text!

Discussion: Which measures can be extracted from text data?

Packages that will be used today

- dplyr
- tidytext
- tidyr
- textdata
- ggplot2
- janeaustenr
- stringr
- syuzhet
- wordcloud
- RColorBrewer

Word Frequency

Let’s create some text in R.

You don’t need to understand this; it is just used for demonstration.

```r
text <- c("Hong Kong Island, known for its dazzling skyline, vibrant culture, and rich history, is a captivating destination nestled on the southeastern coast of China. With its status as a Special Administrative Region, Hong Kong Island stands as a remarkable blend of Eastern and Western influences, creating a unique and dynamic urban landscape. As one of the two main regions that make up the territory of Hong Kong, alongside the Kowloon Peninsula, Hong Kong Island is renowned for its cosmopolitan atmosphere, bustling streets, and iconic landmarks.",
          "The island's story is deeply intertwined with the history of Hong Kong itself. Originally a sparsely populated area, it gradually transformed into a thriving trading port during the 19th century, attracting merchants from around the world. Today, Hong Kong Island embodies the city's economic prowess, with its central business district serving as a global financial hub and a symbol of its economic significance.",
          "The island's skyline is dominated by towering skyscrapers that showcase architectural marvels, blending modernity with traditional Chinese motifs. Among the most prominent landmarks is the famous Hong Kong Convention and Exhibition Centre, an architectural gem situated on the waterfront. The towering Bank of China Tower and the striking International Finance Centre further contribute to the island's impressive skyline.",
          "Beyond its urban splendor, Hong Kong Island offers a diverse range of experiences. The district of Central is a paradise for shopaholics, boasting luxury boutiques, international brands, and bustling street markets. The vibrant neighborhoods of Sheung Wan and Sai Ying Pun offer a glimpse into Hong Kong's colonial past with their quaint streets, antique shops, and historic temples.")
```

Next, we transform the vector into a data frame.

```r
library(dplyr)
library(tidytext)
text_df <- data_frame(line = 1:4, text = text)
text_df
```

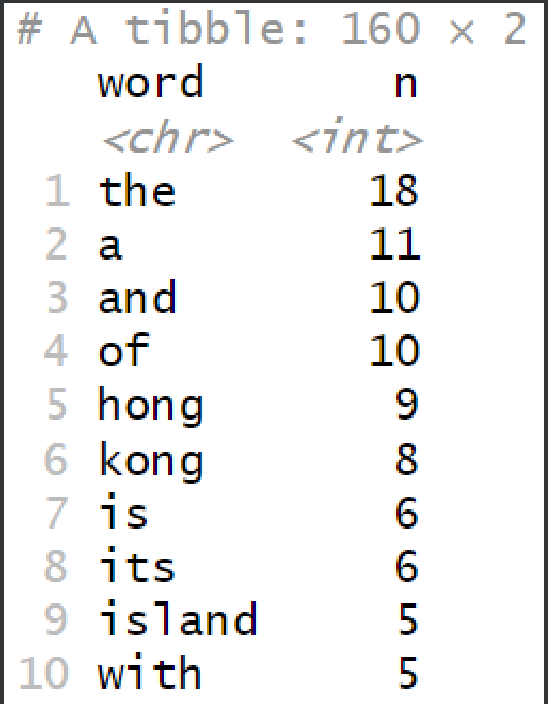
Next, we “tokenize” the text, i.e., breaking down a sequence of text into smaller units called tokens.

```r
mytext <- text_df %>% unnest_tokens(word, text)
```

Here, *%>%* is an operation in the dplyr package, which applies the tokenization function to the text_df data frame.

Then, we count the most frequent words in our input text.

```r
mytext %>% count(word, sort = TRUE)
```

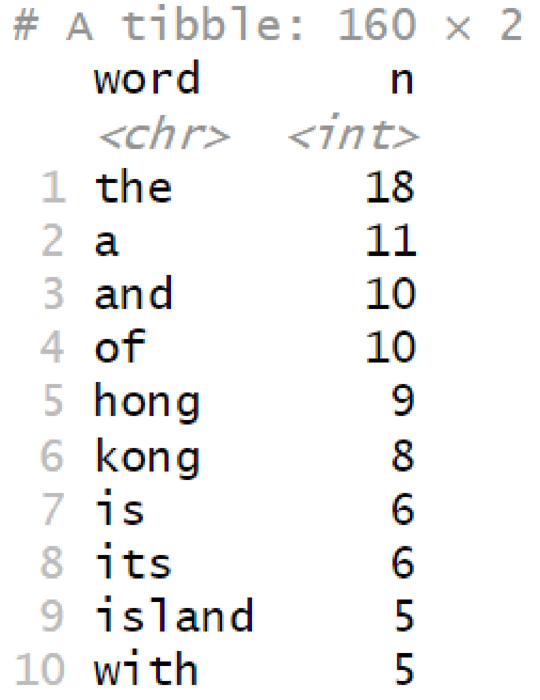


```
# A tibble: 160 × 2
   word       n
   <chr>  <int>
 1 the       18
 2 a         11
 3 and       10
 4 of        10
 5 hong       9
 6 kong       8
 7 is         6
 8 its        6
 9 island     5
10 with       5
```

Any issues with the analysis so far?

Then, we count the most frequent words in our input text.

```
mytext %>% count(word, sort = TRUE)
```



```
# A tibble: 160 × 2
   word       n
   <chr>  <int>
 1 the       18
 2 a         11
 3 and       10
 4 of        10
 5 hong       9
 6 kong       8
 7 is         6
 8 its        6
 9 island     5
10 with       5
```

Any issues with the analysis so far?

The issue is many of the frequent words are meaningless ones such as “the”, “a”. These words appear in any text and do not carry specific meaning. They are called “stop words” in English.

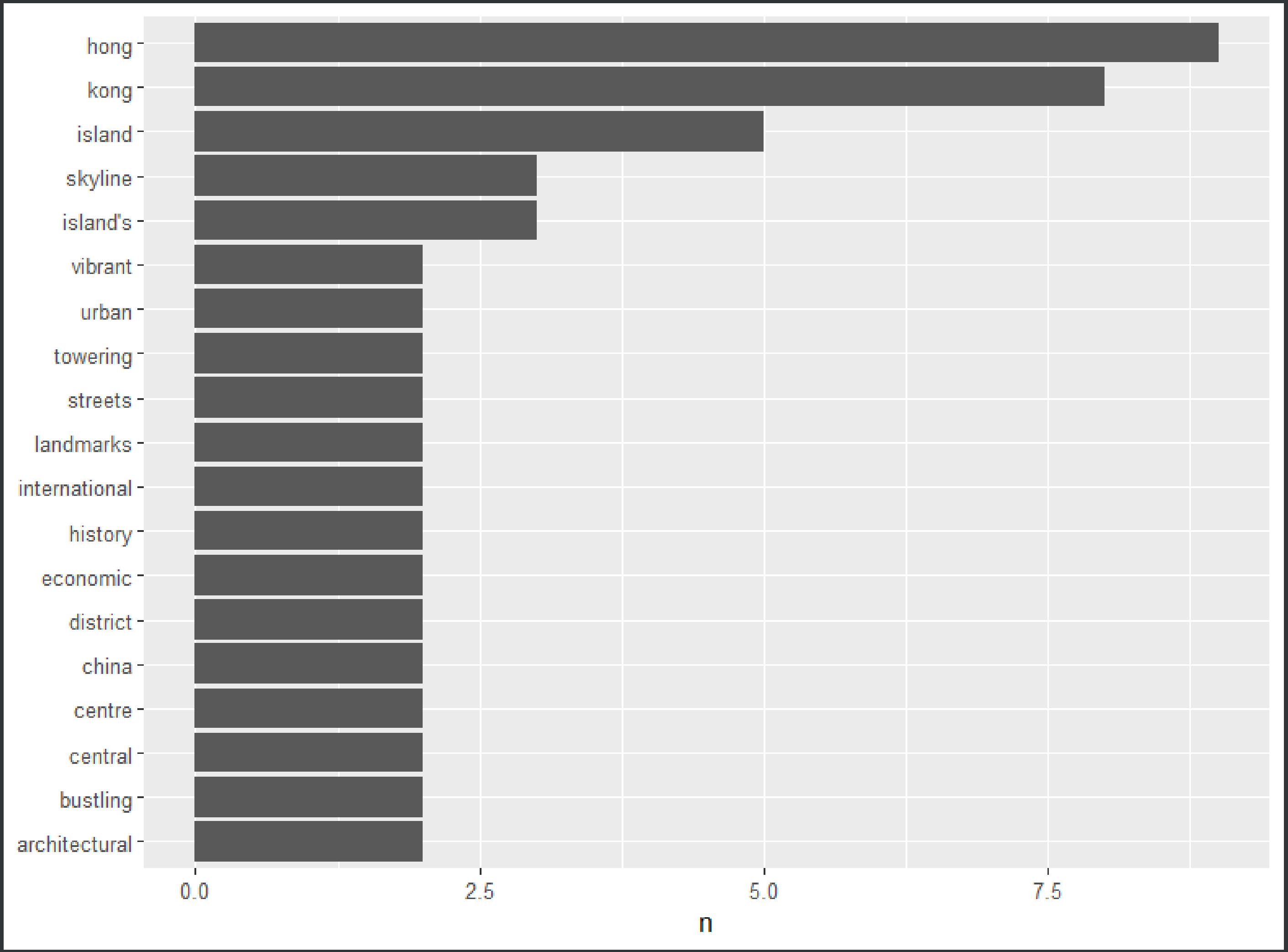
We want to remove the stop words from the text.

```r
data(stop_words)
mytext <- mytext %>% anti_join(stop_words)
mytext %>% count(word, sort = TRUE)
```

Try the code yourself and see if it makes sense to you!

Let’s visualize the most frequent words!

```r
library(ggplot2)
mytext %>%
  count(word, sort = TRUE) %>%     #sorting the words based on frequency
  filter(n > 1) %>%                #display words that appear more than once
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```

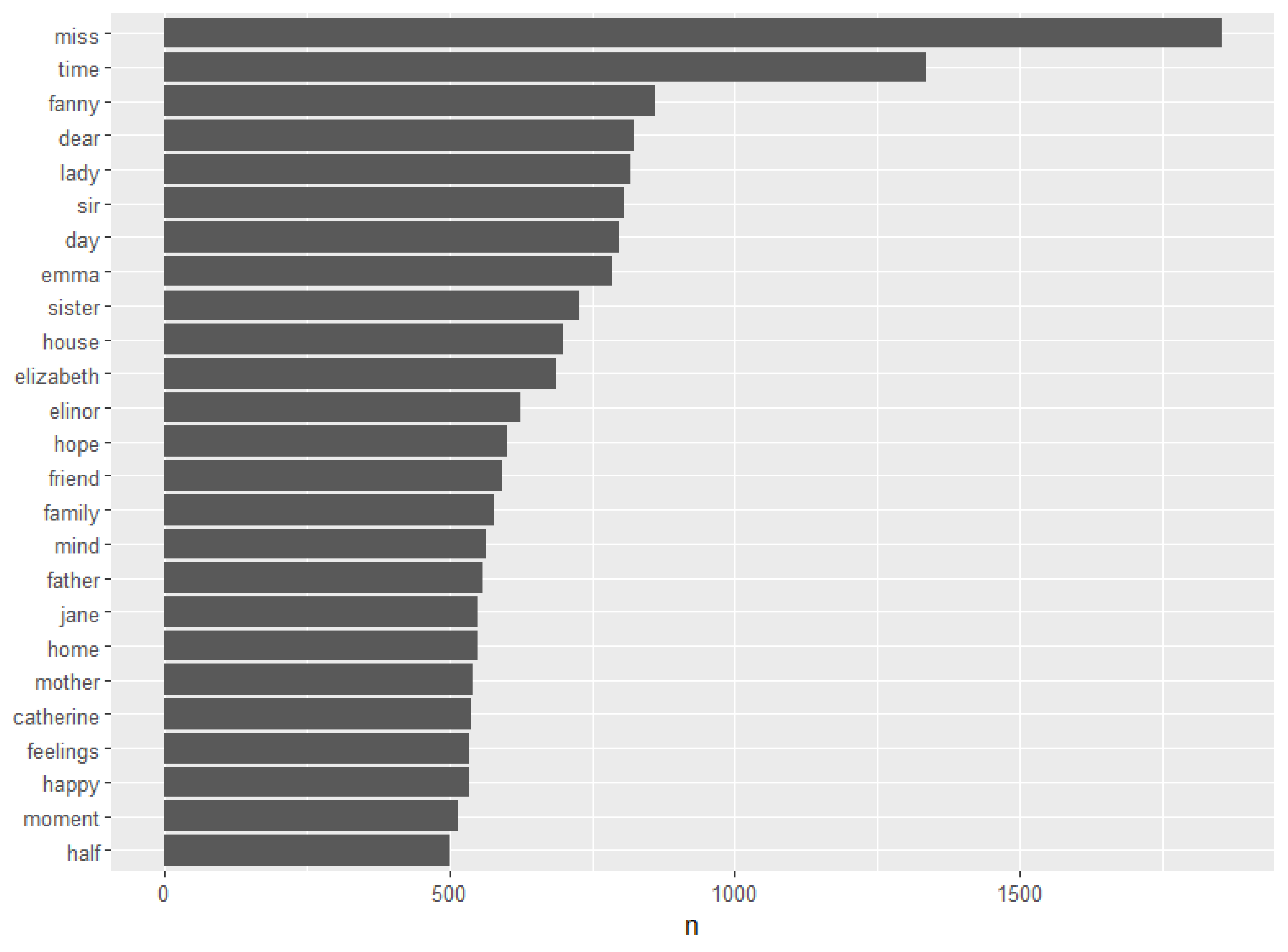

hong
kong
island
skyline
island's
vibrant
urban
towering
streets
landmarks
international
history
economic
district
china
centre
central
bustling
architectural
0.0
2.5
5.0
7.5
n

The complete code is here.

```r
library(dplyr)
library(tidytext)
library(ggplot2)
data(stop_words)
text_df <- data_frame(line = 1:4, text = text)
mytext <- text_df %>% unnest_tokens(word, text)
mytext <- mytext %>% anti_join(stop_words)
mytext %>%
  count(word, sort = TRUE) %>%    #sorting the words based on frequency
  filter(n > 1) %>%               #display words that appear more than once
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```

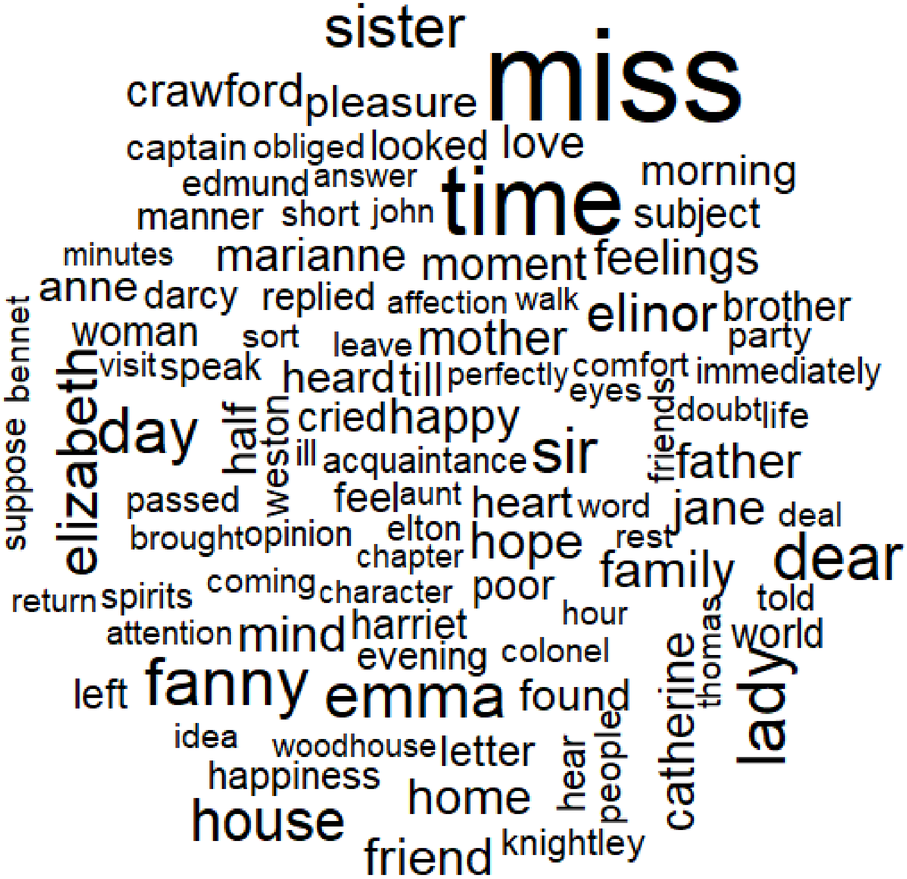
In another example, we analyze the distribution of most frequent words in works of Jane Austen

```r
library(dplyr)
library(tidytext)
library(ggplot2)
library(janeaustenr)
data(stop_words)
original_books <- austen_books()
mytext <- original_books %>%
  unnest_tokens(word, text)
mytext <- mytext %>% anti_join(stop_words)
mytext %>%
  count(word, sort = TRUE) %>%
  filter(n > 500) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```


miss
time
fanny
dear
lady
sir
day
emma
sister
house
elizabeth
elinor
hope
friend
family
mind
father
jane
home
mother
catherine
feelings
happy
moment
half
0
500
1000
1500
n

We can further generate a word cloud of the frequent words:

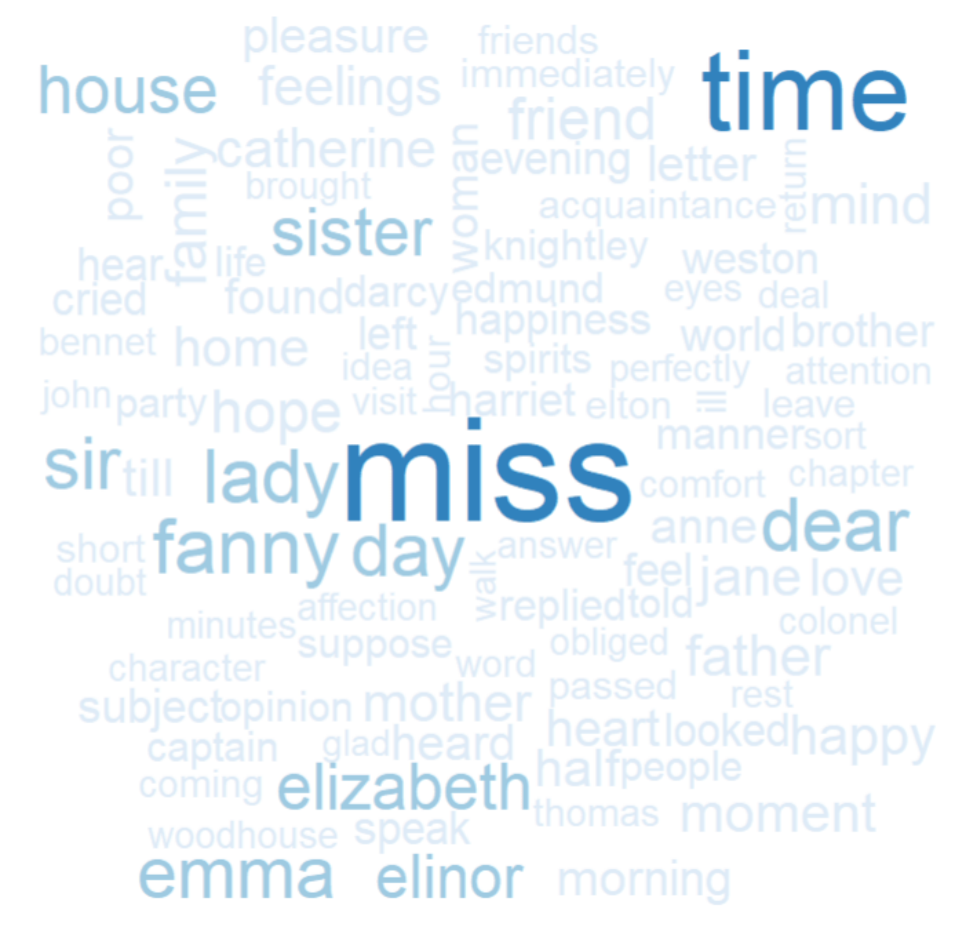
```r
library(wordcloud)
mytext %>%
 anti_join(stop_words) %>%
 count(word) %>%
 with(wordcloud(word, n, max.words = 100))
```


sister
miss
crawford
pleasure
captain
obliged
looked
love
edmund
answer
time
morning
manner
short
john
subject
minutes
marianne
moment
feelings
anne
darcy
replied
affection
walk
elinor
brother
bennet
woman
sort
leave
mother
party
elizabeth
visit
speak
heard
till
perfectly
comfort
immediately
eyes
suppose
day
half
weston
cried
happy
friends
doubt
life
ill
acquaintance
sir
father
passed
feel
aunt
heart
word
jane
deal
brought
opinion
elton
chapter
hope
rest
dear
coming
family
return
spirits
character
poor
told
hour
attention
mind
harriet
thomas
world
evening
colonel
catherine
left
fanny
emma
found
lady
idea
woodhouse
letter
hear
people
happiness
home
house
knightley
friend

Adding colors to the word cloud:

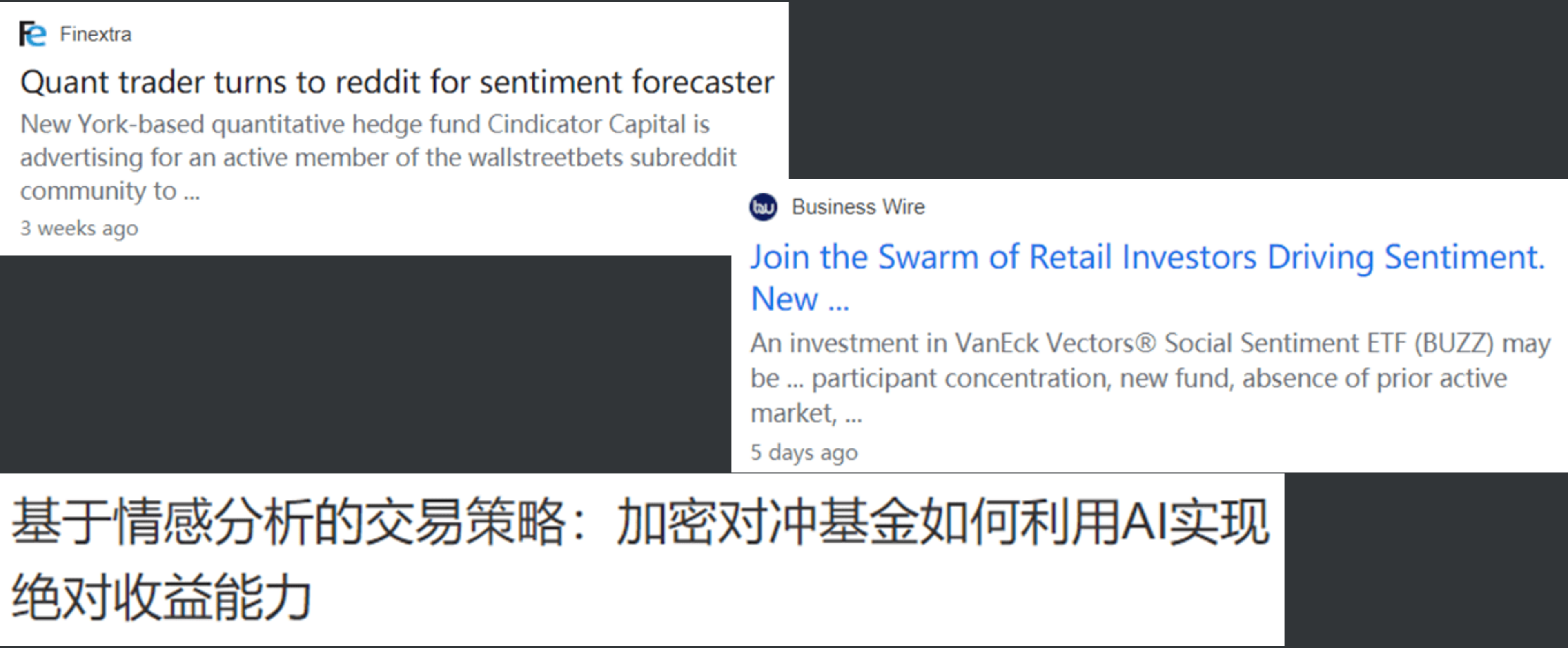
```r
library(wordcloud)
library(RColorBrewer)

mytext %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100,
                 colors = brewer.pal(3, "Blues")))
```

We choose three blue colors.


pleasure
friends
house
feelings
immediately
time
friend
poor
family
catherine
woman
evening
letter
return
brought
mind
acquaintance
sister
knightley
weston
hear
life
cried
found
darcy
edmund
eyes
deal
happiness
bennet
home
left
world
brother
idea
hour
spirits
perfectly
attention
john
party
hope
visit
harriet
elton
ill
leave
manner
sort
sir
till
lady
miss
comfort
chapter
anne
dear
short
fanny
day
answer
walk
feel
doubt
jane
love
affection
replied
told
minutes
colonel
suppose
obliged
word
father
character
passed
rest
subject
opinion
mother
heart
looked
happy
captain
glad
heard
half
people
coming
elizabeth
thomas
moment
woodhouse
speak
emma
elinor
morning

Sentiment Analysis

Sentiment Analysis is arguably the most important type of text analysis. Basically, we want to classify text based on the *valence,* which can be either positive or negative (sometimes it can also be neutral).



Finextra

Quant trader turns to reddit for sentiment forecaster

New York-based quantitative hedge fund Cindicator Capital is advertising for an active member of the wallstreetbets subreddit community to ...

3 weeks ago

Business Wire

Join the Swarm of Retail Investors Driving Sentiment. New ...

An investment in VanEck Vectors® Social Sentiment ETF (BUZZ) may be ... participant concentration, new fund, absence of prior active market, ...

5 days ago

基于情感分析的交易策略：加密对冲基金如何利用AI实现绝对收益能力

Question: In your own opinion, how should we perform sentiment analysis?

Sentiment Analysis

The basic idea of sentiment analysis is rather simple.

We can build two lexicons (i.e., dictionaries) of positive and negative words. Here are some examples:

- Positive: great, amazing, fantastic, excel, …
- Negative: ugly, terrible, awful, failed, …

Click here to see examples of sentiment lexicons.

Sentiment Analysis

Naturally, if a sentence contains more positive words, it likely expresses some positive feeling. Instead, a sentence containing many negative words are likely to express a negative emotion.

In addition, some words are “more positive” than others. For example, “great” and “awesome” are stronger than “OK” and “so so”. In this case, we can assign different weights to different words.

Simple Methods

The syuzhet package is a simple R package which allows you to perform sentiment analysis.

```r
library("syuzhet")
text = "HKU is a fantastic school, I love it."
syuzhet_vector <- get_sentiment(text, method="syuzhet")
head(syuzhet_vector)
```

A positive (negative) output implies positive (negative) sentiment.

Simple Methods

You can perform sentiment analysis with other lexicons.

```r
text = "HKU is a nice school and I like it."
bing_vector <- get_sentiment(text, method="bing")
head(bing_vector)

afinn_vector <- get_sentiment(text, method="afinn")
head(afinn_vector)
```

Simple Methods

We can go beyond sentiment analysis to find out other emotions as well.

```
text = "HKU is a terrible school."
print(get_nrc_sentiment(text))
```

The Sentiments Dataset

The tidytext package contains several sentiment lexicons in the sentiments dataset. To visualize the datasets, try the following code:

```
library(tidytext)
library(textdata)
sentiments
```

The Sentiments Dataset

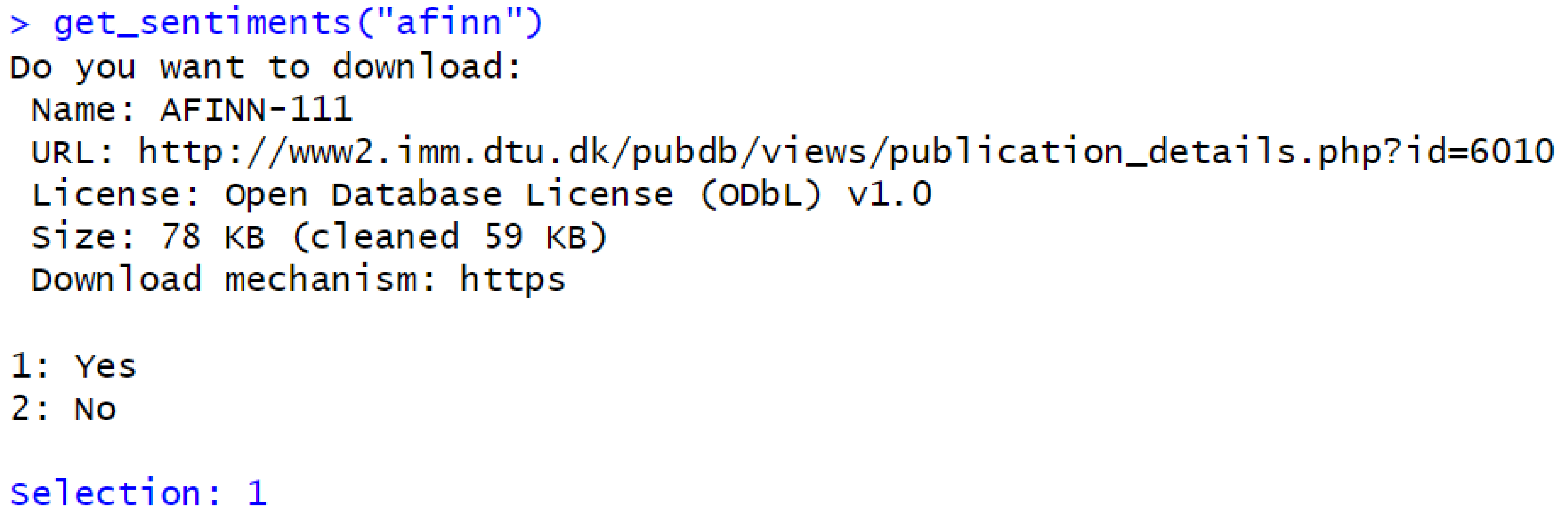
The tidytext package contains three general-purpose lexicons, namely

- AFINN from Finn Årup Nielsen
- Bing from Bing Liu and collaborators
- NRC from Saif Mohammad and Peter Turney

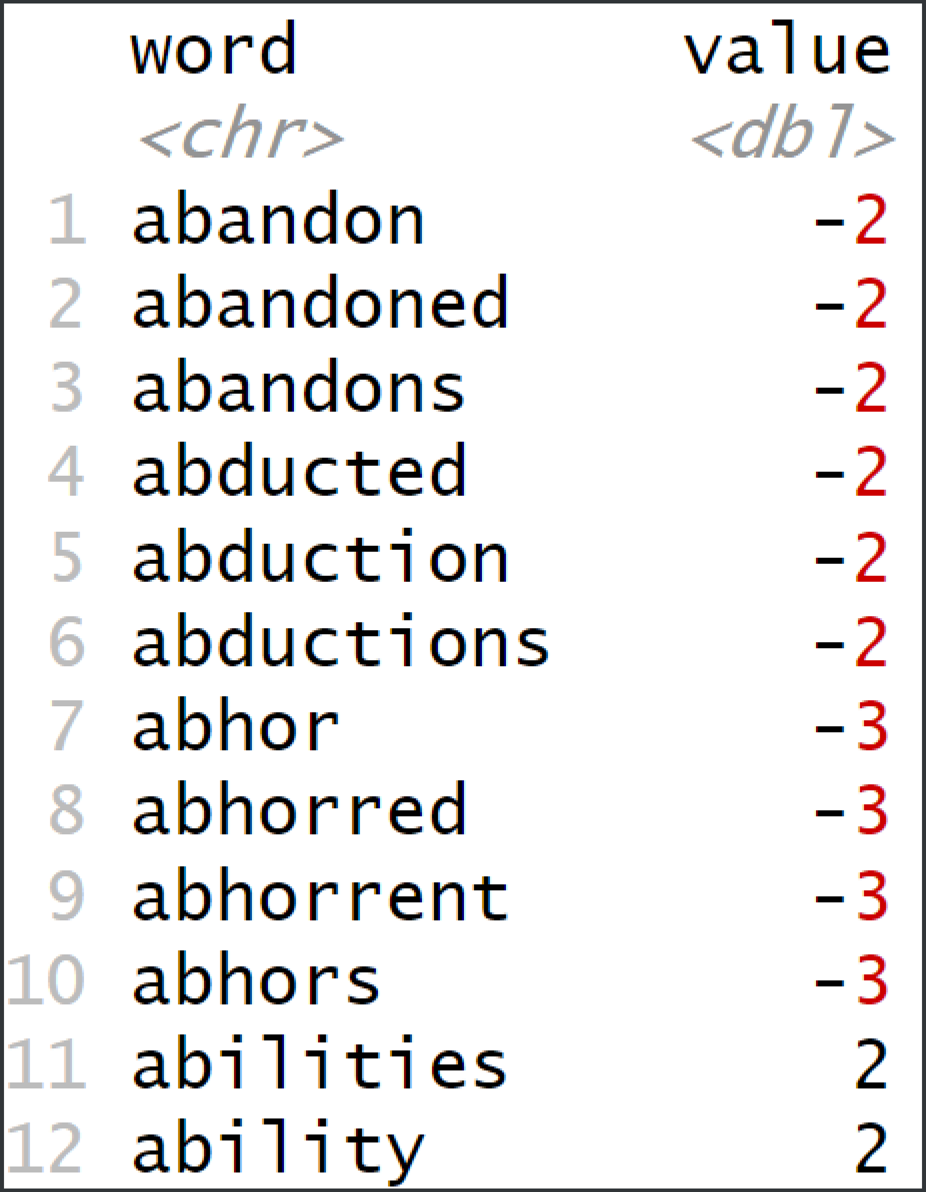
View the lexicons:

```r
library(tidytext)
library(textdata)
get_sentiments("afinn")
afinn <- get_sentiments("afinn")
print(afinn, n = 100)
```

Enter “1” to download the lexicons.

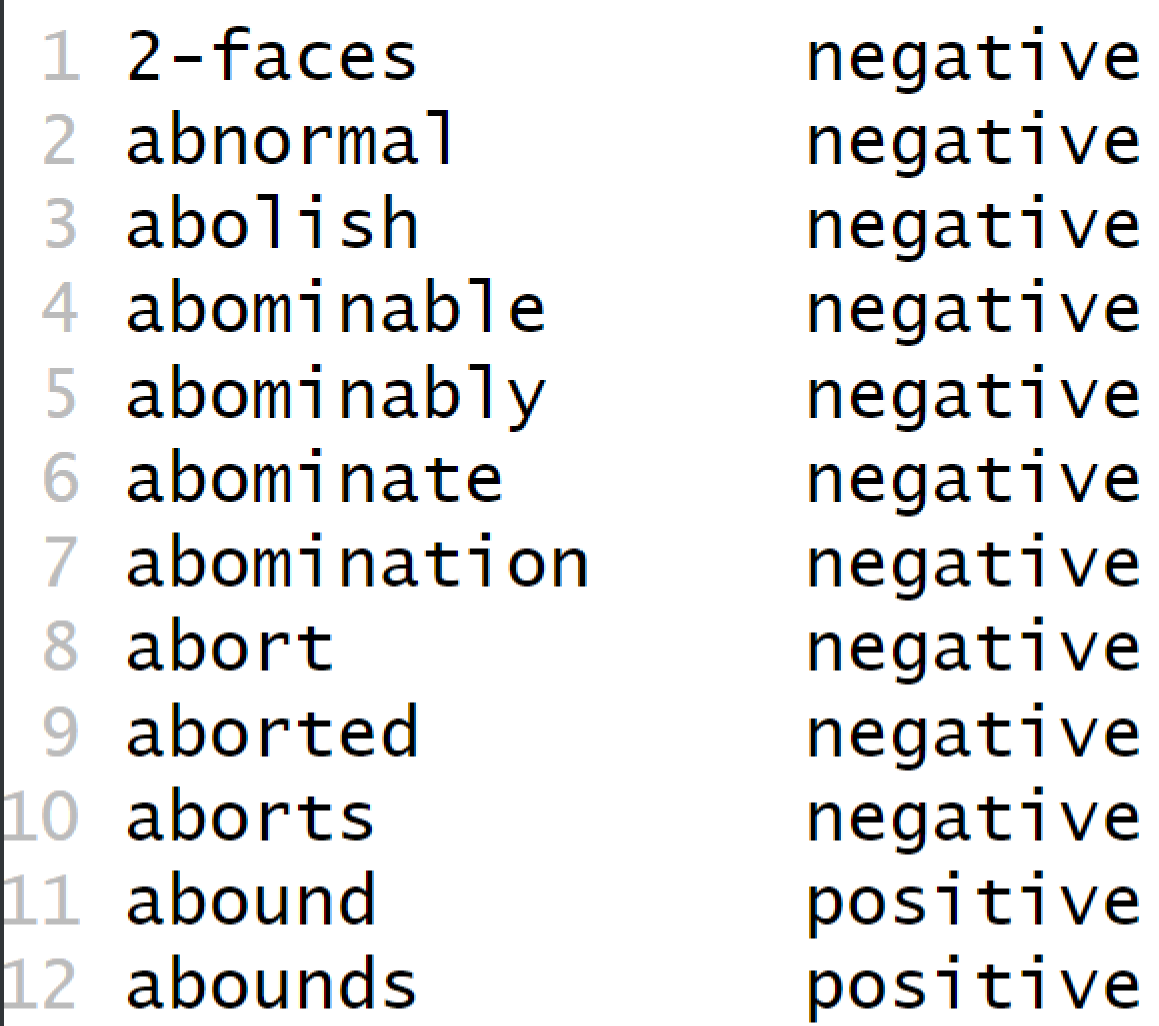


```
> get_sentiments("afinn")
Do you want to download:
 Name: AFINN-111
 URL: http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
 License: Open Database License (ODbL) v1.0
 Size: 78 KB (cleaned 59 KB)
 Download mechanism: https

1: Yes
2: No

Selection: 1
```



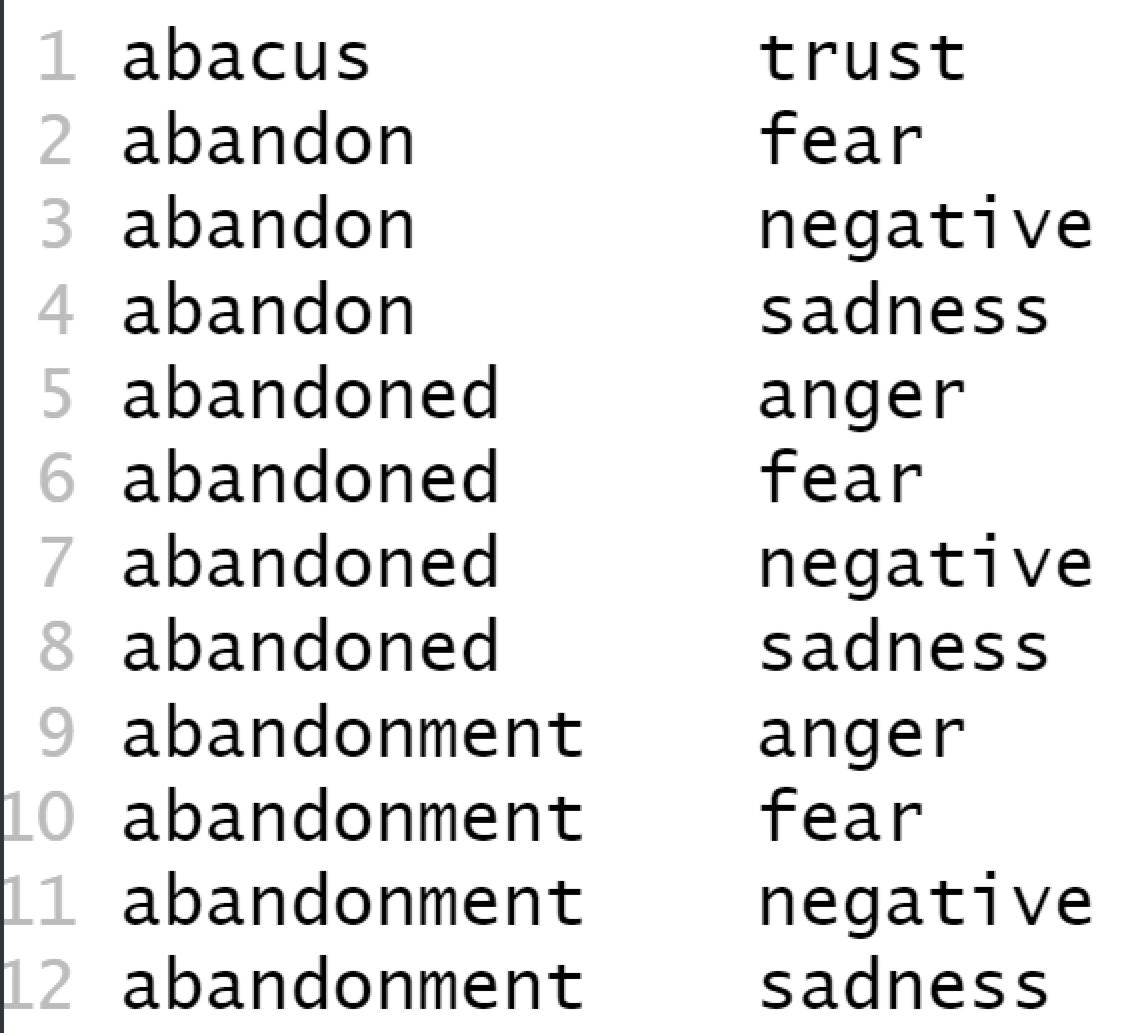
```
   word        value
   <chr>       <dbl>
 1 abandon        -2
 2 abandoned      -2
 3 abandons       -2
 4 abducted       -2
 5 abduction      -2
 6 abductions     -2
 7 abhor          -3
 8 abhorred       -3
 9 abhorrent      -3
10 abhors         -3
11 abilities       2
12 ability         2
```

```r
bing <- get_sentiments("bing")
print(bing, n = 100)
```



```
 1 2-faces      negative
 2 abnormal     negative
 3 abolish      negative
 4 abominable   negative
 5 abominably   negative
 6 abominate    negative
 7 abomination  negative
 8 abort        negative
 9 aborted      negative
10 aborts       negative
11 abound       positive
12 abounds      positive
```

```r
nrc <- get_sentiments("nrc")
print(nrc, n = 100)
```



```
 1 abacus         trust
 2 abandon        fear
 3 abandon        negative
 4 abandon        sadness
 5 abandoned      anger
 6 abandoned      fear
 7 abandoned      negative
 8 abandoned      sadness
 9 abandonment    anger
10 abandonment    fear
11 abandonment    negative
12 abandonment    sadness
```

A Chinese lexicon here

Which words reflect “joy” in Jane Austen’s books?

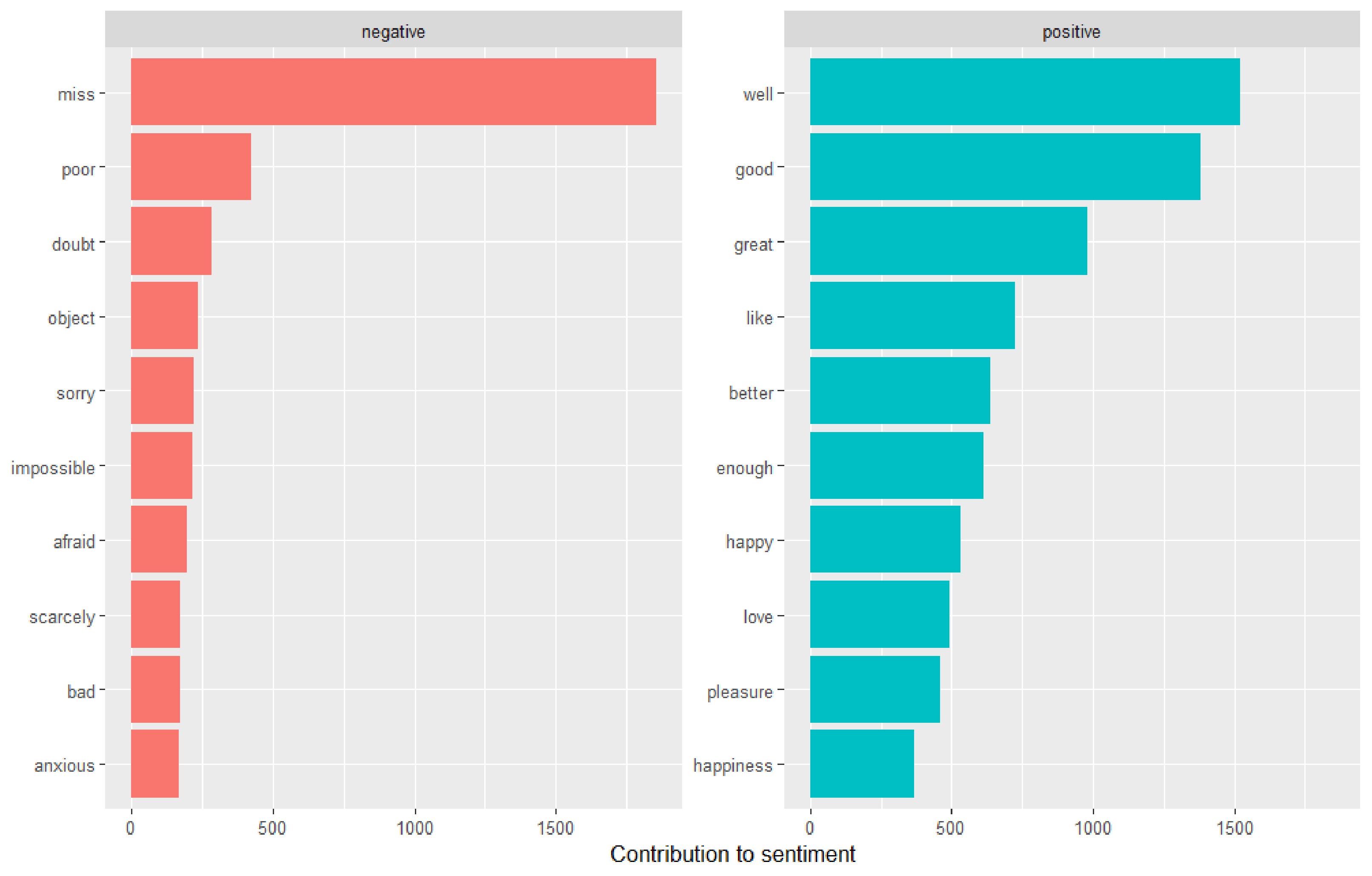
```r
library(janeaustenr)
library(dplyr)
library(stringr)
tidy_books <- austen_books()  %>%
unnest_tokens(word, text)

nrcjoy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

tidy_books %>%
  inner_join(nrcjoy) %>%
  count(word, sort = TRUE)
```

Most common positive/negative words

```r
bing_word_counts <- tidy_books %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE)

bing_word_counts
```

Visualizing top positive/negative words

```r
bing_word_counts %>%
  group_by(sentiment) %>%
  slice_max(n, n = 10) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(x = "Contribution to sentiment",
       y = NULL)
```


negative
positive
miss
poor
doubt
object
sorry
impossible
afraid
scarcely
bad
anxious
well
good
great
like
better
enough
happy
love
pleasure
happiness
0
500
1000
1500
0
500
1000
1500
Contribution to sentiment

Question: Are there any issues with the lexicon based approach?

Issue #1: Negations

- I don't think HKU is a great school.
- None of us like HKU.

Solution: When we see "not", we take the opposite of every word after it until we reach the first punctuation.

I did not like this movie, but I enjoyed this cinema…

Issue #2: Phrases

- Installing Tableau Public is a piece of cake.
- Buying the new iPhone costs me an arm and a leg.

Solution: Adding phrases to your lexicon as well.

Issue #3: Sarcasm

- This phone has an awesome battery back-up of 38 hours.
- This phone has an awesome battery back-up of 2 hours.

- It's +25 outside and I am so hot.
- It's -25 outside and I am so hot.

- This is the best laptop bag ever. It is so good that within two months of use, it is worthy of being used as a grocery bag.

Issue #4: Comparison

- This product is second to none.
- This one is better than the old one.
- This one is better than nothing.

Issue #5: Objective Sentences

- I followed his recommendation and bought the stock at $200. Now the stock price is $100.
- He started learning R last year. Today, he does not know how to run a linear regression.

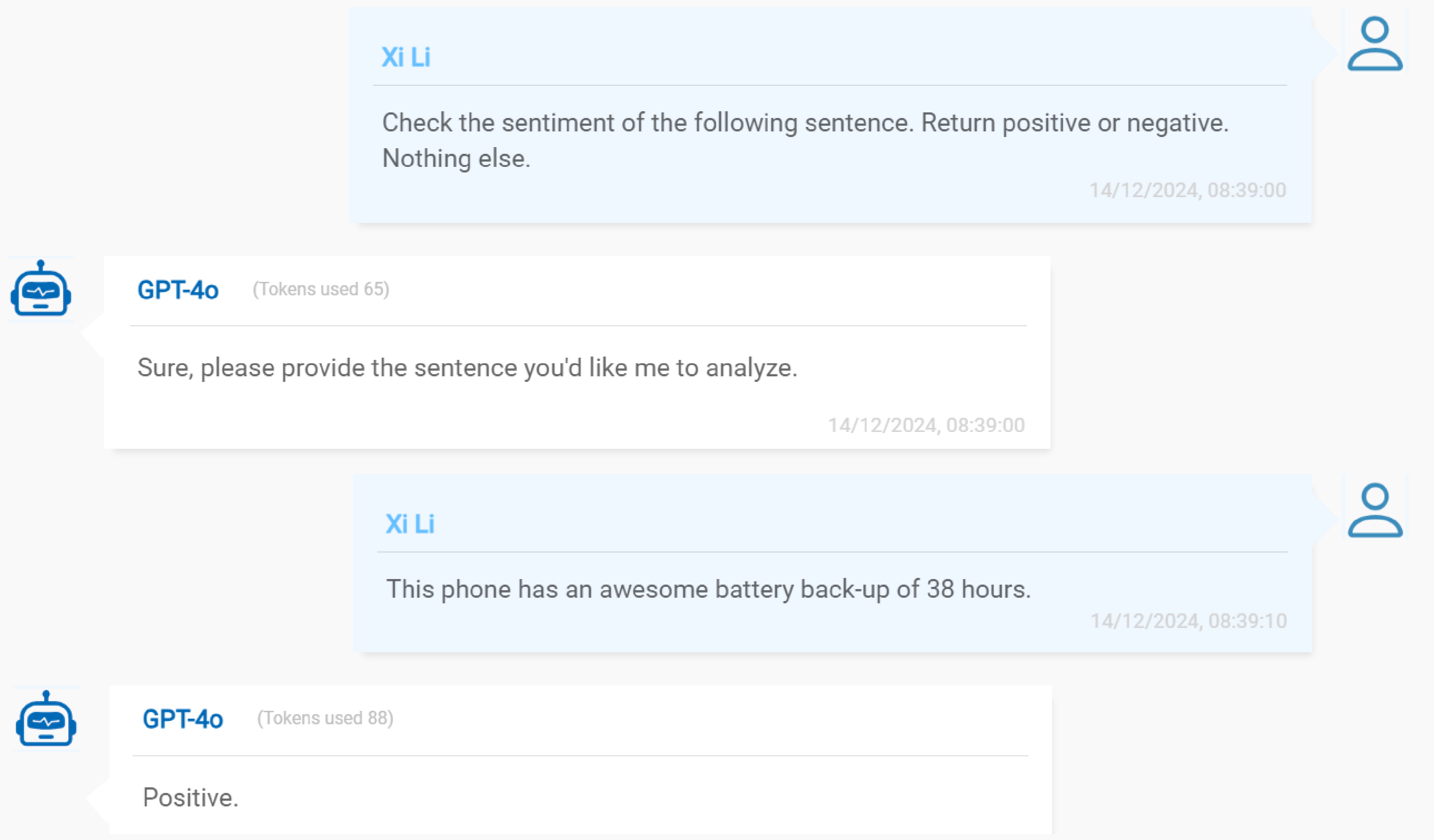
Issue #6: Intensions

- I am going to throw my Lenovo laptop out of the window.
- I am returning this table to IKEA tomorrow.

Issue #7: Indirect Opinions

- With R, I can complete all data analysis in 1 hour that used to take me 3 hours in the past.
- After getting the new lenses, I am able to drive at night again.

https://www.youtube.com/embed/si8zZHkufRY?enablejsapi=1



Xi Li

Check the sentiment of the following sentence. Return positive or negative. Nothing else.

14/12/2024, 08:39:00

GPT-4o (Tokens used 65)

Sure, please provide the sentence you'd like me to analyze.

14/12/2024, 08:39:00

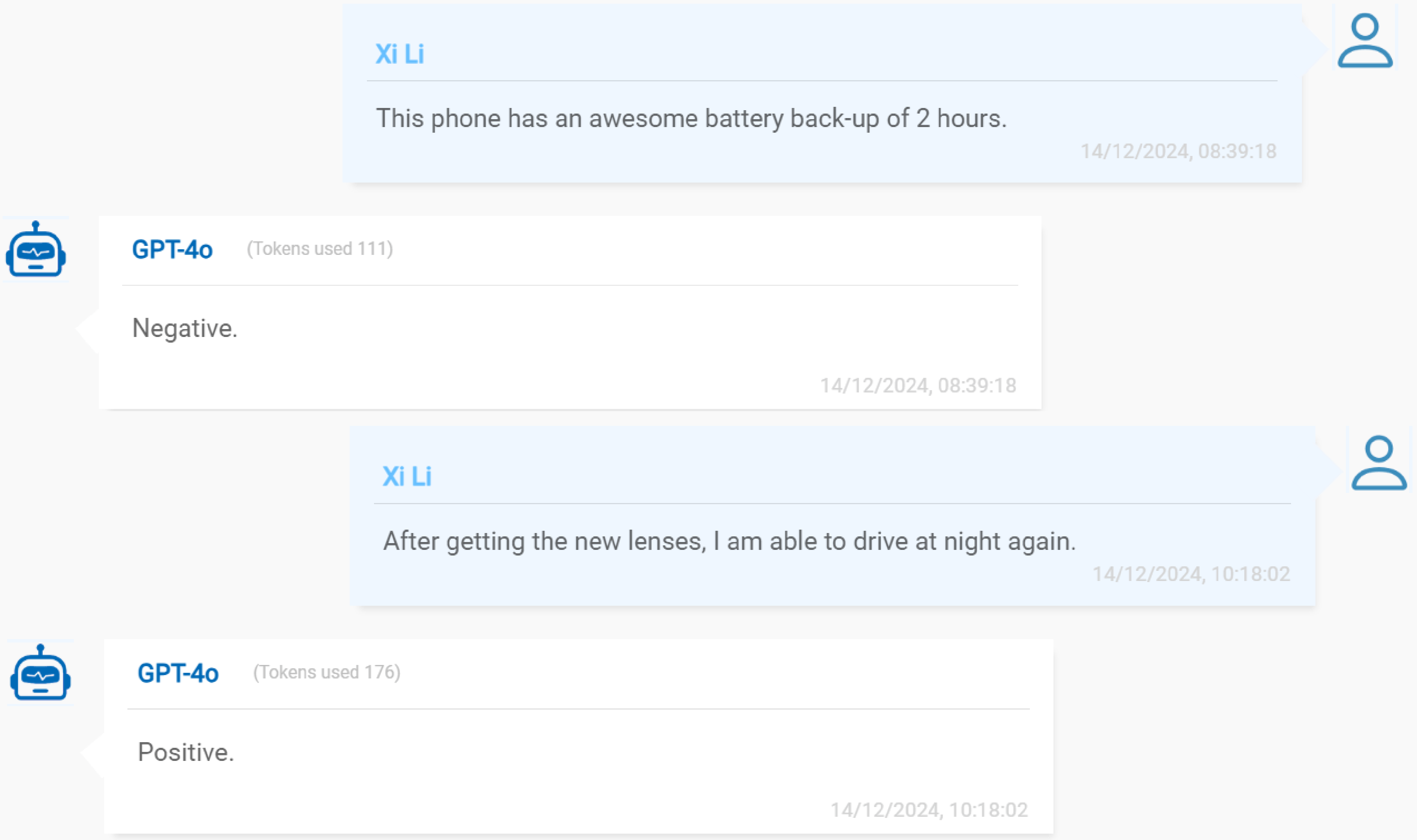
Xi Li

This phone has an awesome battery back-up of 38 hours.

14/12/2024, 08:39:10

GPT-4o (Tokens used 88)

Positive.



Xi Li

This phone has an awesome battery back-up of 2 hours.

14/12/2024, 08:39:18

GPT-4o (Tokens used 111)

Negative.

14/12/2024, 08:39:18

Xi Li

After getting the new lenses, I am able to drive at night again.

14/12/2024, 10:18:02

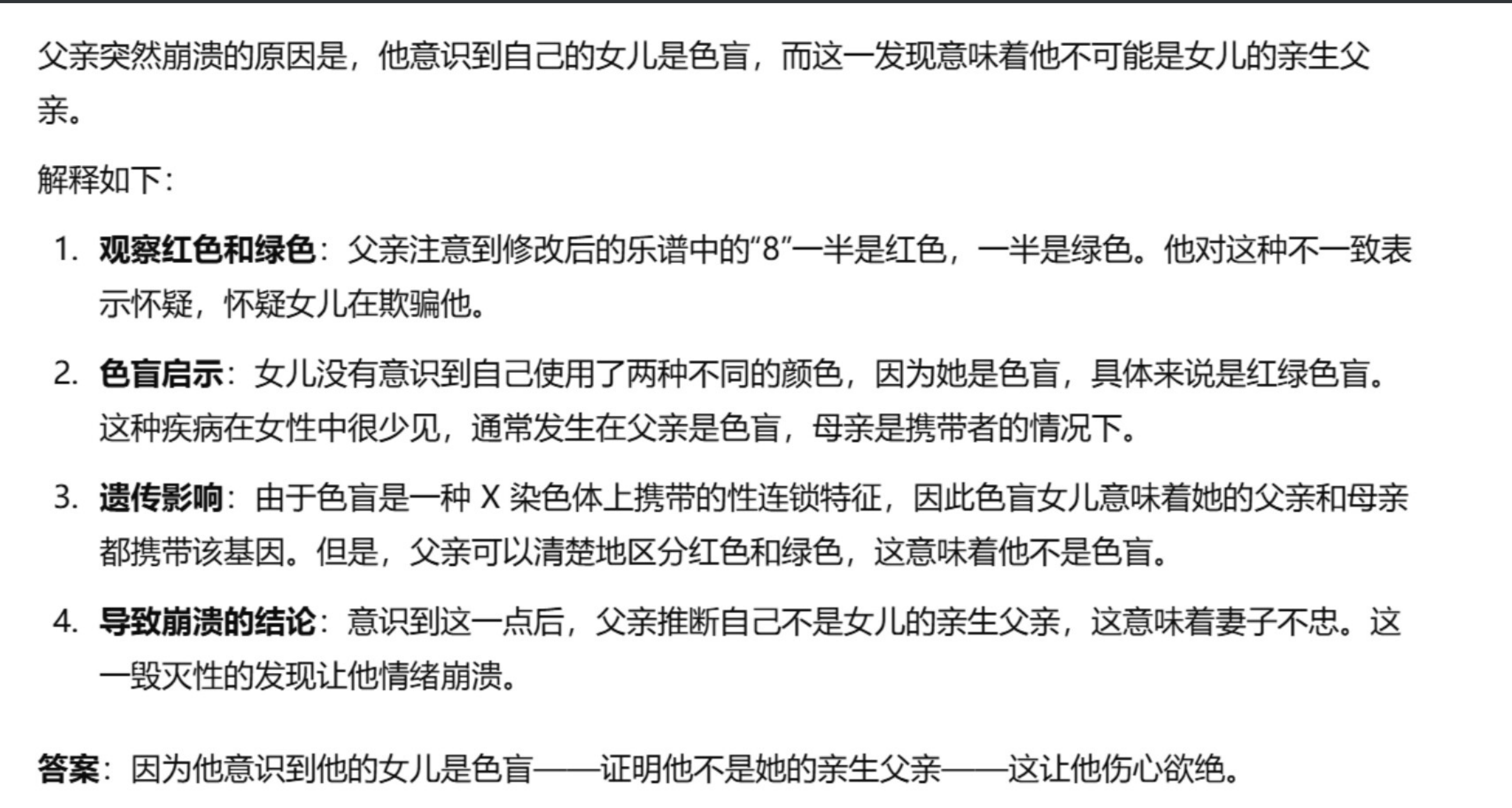
GPT-4o (Tokens used 176)

Positive.

14/12/2024, 10:18:02

有一天，一个女孩参加数学考试只得了38分。她心里对父亲的惩罚充满恐惧，于是偷偷把分数改成了88分。她的父亲看到试卷后，怒发冲冠，狠狠地给了她一巴掌，怒吼道：“你这8怎么一半是绿的一半是红的，你以为我是傻子吗？”女孩被打后，委屈地哭了起来，什么也没说。过了一会儿，父亲突然崩溃了 。父亲崩溃的原因是什么？

One day, a girl took a math test and scored only 38 points. Filled with fear of her father’s punishment, she secretly changed the score to 88. After seeing the test paper, her father was furious, gave her a harsh slap, and roared: “Why is half of this 8 green and the other half red, do you think I’m a fool?” After being slapped, the girl cried grievously and said nothing. After a while, the father suddenly broke down. What was the reason for the father's breakdown?



父亲突然崩溃的原因是，他意识到自己的女儿是色盲，而这一发现意味着他不可能是女儿的亲生父亲。

解释如下：

1. **观察红色和绿色**：父亲注意到修改后的乐谱中的“8”一半是红色，一半是绿色。他对这种不一致表示怀疑，怀疑女儿在欺骗他。
2. **色盲启示**：女儿没有意识到自己使用了两种不同的颜色，因为她是色盲，具体来说是红绿色盲。这种疾病在女性中很少见，通常发生在父亲是色盲，母亲是携带者的情况下。
3. **遗传影响**：由于色盲是一种 X 染色体上携带的性连锁特征，因此色盲女儿意味着她的父亲和母亲都携带该基因。但是，父亲可以清楚地区分红色和绿色，这意味着他不是色盲。
4. **导致崩溃的结论**：意识到这一点后，父亲推断自己不是女儿的亲生父亲，这意味着妻子不忠。这一毁灭性的发现让他情绪崩溃。

答案：因为他意识到他的女儿是色盲——证明他不是她的亲生父亲——这让他伤心欲绝。

The reason for the father’s breakdown is that he realized his daughter is colorblind, which implies that he cannot be her biological father. This realization devastated him. Explanation:

- Observing the red and green colors: The father noticed that the modified score of "88" had one half in red and the other half in green. He was suspicious of this inconsistency and suspected his daughter was trying to deceive him.
- Indication of color blindness: The daughter did not realize she used two different colors because she is colorblind, specifically red-green colorblind. This condition is rare in females and usually occurs when the father is colorblind, and the mother is a carrier.
- Genetic influence: Color blindness is an X-linked trait, so a colorblind daughter implies that both her father and mother carry the gene. However, the father can clearly distinguish between red and green, indicating he is not colorblind.
- Conclusion leading to breakdown: Upon realizing this, the father concluded that he is not the biological father of his daughter, which implied his wife’s infidelity. This devastating discovery led to his emotional breakdown.

Answer: The father broke down because he realized his daughter was colorblind, proving he was not her biological father, which devastated him.

n-gram

n-gram

In the analysis so far, we've considered words as individual units, and considered their relationships to sentiments. However, many interesting text analyses are based on the relationships between words, whether examining which words tend to follow others immediately, or that tend to co-occur within the same documents.

n-gram

Consider the following text: “The University of Hong Kong is the oldest institution in Hong Kong.” In the text, we have tokens such as “University”, “institution”. In addition to that, we have consecutive sequences of words, which are known as n-gram:

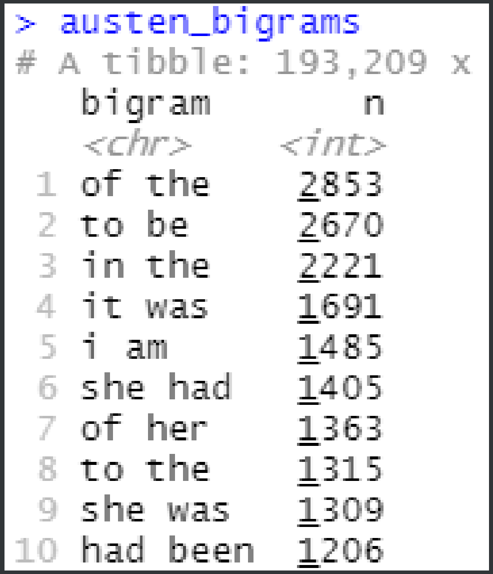
- bigram ($n = 2$): Hong Kong, oldest institution, ...
- trigram ($n = 3$): University of Hong Kong, the oldest institution, ...

n-gram

```r
library(dplyr)
library(tidytext)
library(janeaustenr)
library(tidyr)


austen_bigrams <- austen_books() %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)  %>%
  count(bigram, sort = TRUE) %>%
  filter(!is.na(bigram))

austen_bigrams
```

Here, we create bigrams from Jane Austen’s books, sort them by frequency, and remove empty bigrams.

n-gram



```
> austen_bigrams
# A tibble: 193,209 x
   bigram        n
   <chr>     <int>
 1 of the     2853
 2 to be      2670
 3 in the     2221
 4 it was     1691
 5 i am       1485
 6 she had    1405
 7 of her     1363
 8 to the     1315
 9 she was    1309
10 had been   1206
```

All of these bigrams are meaningless!

Next Step: Remove bigrams containing stop words.

n-gram

```r
bigrams_separated <- austen_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

bigrams_filtered
```

The complete code is here.

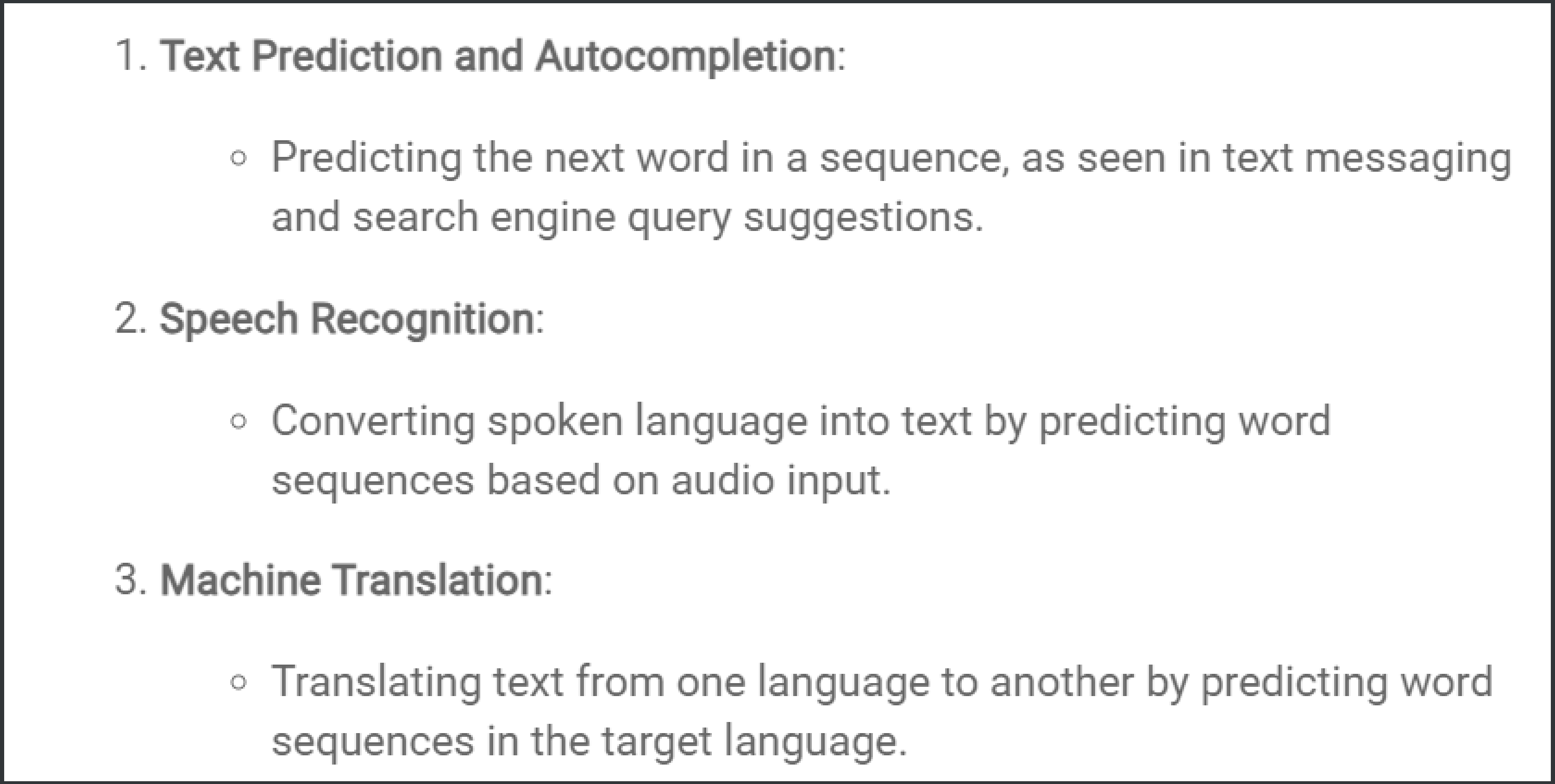
```r
library(dplyr)
library(tidytext)
library(janeaustenr)
library(tidyr)


austen_bigrams <- austen_books() %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)  %>%
  count(bigram, sort = TRUE) %>%
  filter(!is.na(bigram))

bigrams_separated <- austen_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

bigrams_filtered
```

How about trigrams?

```r
austen_books() %>%
  unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
  filter(!is.na(trigram)) %>%
  separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word,
         !word3 %in% stop_words$word) %>%
  count(word1, word2, word3, sort = TRUE)
```

What are the applications of the n-gram model?

I asked GPT...



1. **Text Prediction and Autocompletion**:
   - Predicting the next word in a sequence, as seen in text messaging and search engine query suggestions.
2. **Speech Recognition**:
   - Converting spoken language into text by predicting word sequences based on audio input.
3. **Machine Translation**:
   - Translating text from one language to another by predicting word sequences in the target language.

Topic Models

Topic Models

In text mining, we often have collections of documents, such as blog posts or news articles, that we'd like to divide into natural groups so that we can understand them separately.

Topic Models

In your opinion, what is a topic?

Topic Models

A topic is a distribution over words. For example, when you mention the following words frequently, you are likely to talk about the topic “marketing”: advertising, customer relationship, distribution channel, pricing, product, sales, market research, …

Topic Models

A topic is a distribution over words. For example, when you mention the following words frequently, you are likely to talk about the topic “flight”: airport, ticket, delay, shuttle bus, check-in, business-class …

Topic Models

How to find out topics from a number of documents?

Topic Models

This is a complex process. The basic idea is, words of the same topic would often appear together. For example, “airport” and “shuttle bus” often appear in the same sentence because they belong to the same topic; “university” and “education” often appear in the same sentence as well. However, “airport” and “education” do not appear frequently in the same sentence.

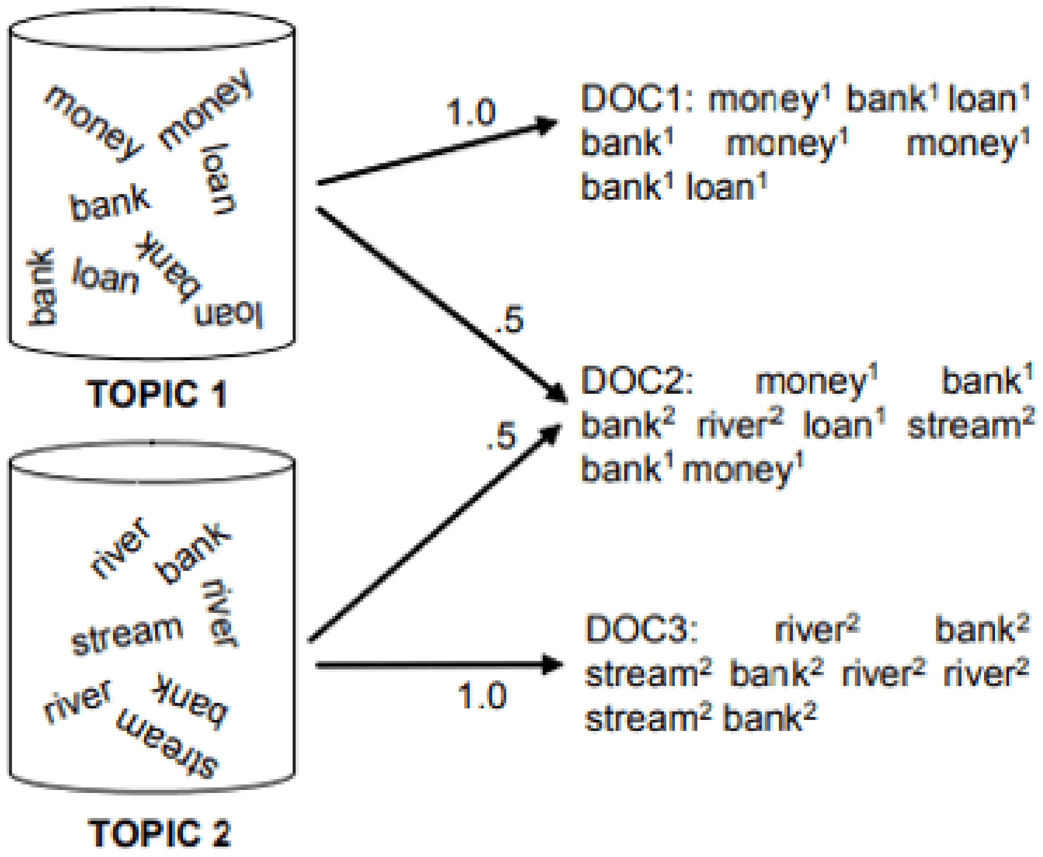
The Basics of Topic Models

Every topic is a mixture of words: For example, we could imagine a two-topic model of news, with one topic for "politics" and one for "entertainment." The most common words in the politics topic might be "President," "Congress," and "government," while the entertainment topic may be made up of words such as "movies," "television," and "actor." Words can be shared between topics; a word like "budget" might appear in both equally.

The Basics of Topic Models

Every document is a mixture of topics: We imagine that each document may contain words from several topics in particular proportions. For example, in a two-topic model we could say "Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B."

PROBABILISTIC GENERATIVE PROCESS



DOC1: money^1 bank^1 loan^1 bank^1 money^1 money^1 bank^1 loan^1

DOC2: money^1 bank^1 bank^2 river^2 loan^1 stream^2 bank^1 money^1

DOC3: river^2 bank^2 stream^2 bank^2 river^2 river^2 stream^2 bank^2

Topics

gene	0.04
dna	0.02
genetic	0.01
...	

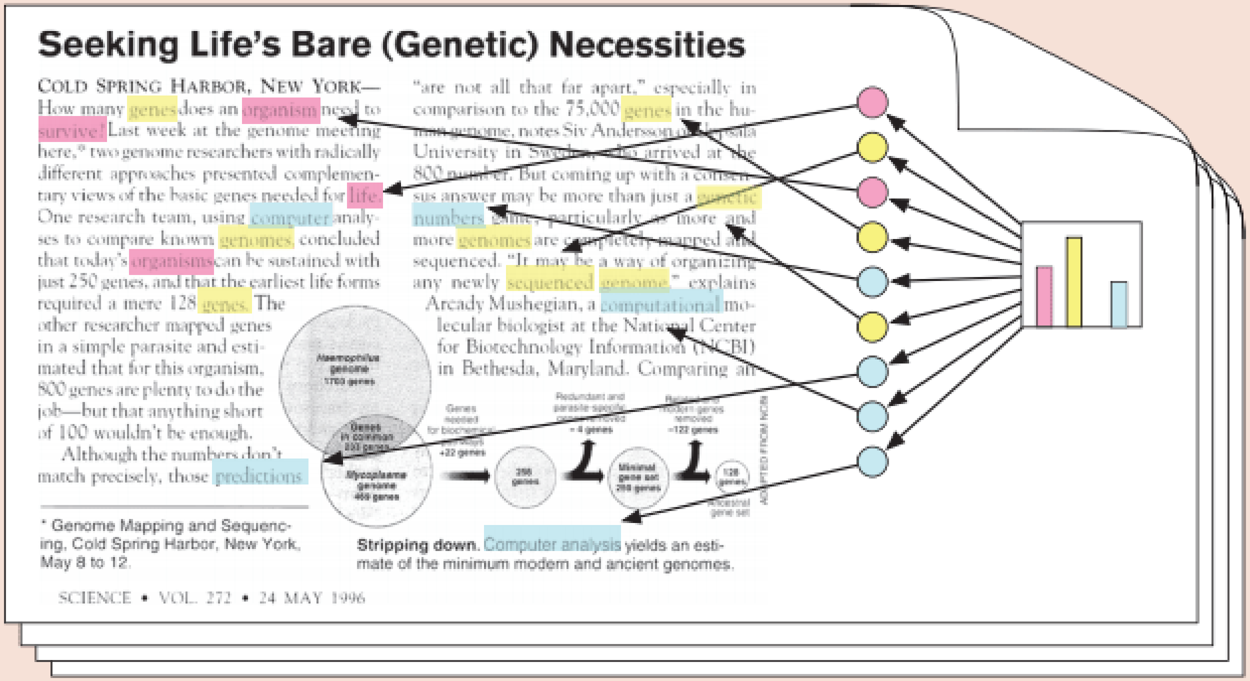
life	0.02
evolve	0.01
organism	0.01
...	

brain	0.04
neuron	0.02
nerve	0.01
...	

data	0.02
number	0.02
computer	0.01
...	

Documents

Topic proportions and assignments



Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Haemophilus genome 1703 genes
Genes in common 233 genes
Mycoplasma genome 469 genes
Genes needed for biochemical pathways +22 genes
256 genes
Redundant and parasite-specific genes removed -4 genes
Minimal gene set 250 genes
Related and modern genes removed -122 genes
128 genes
Ancestral gene set
ADAPTED FROM NCBI

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

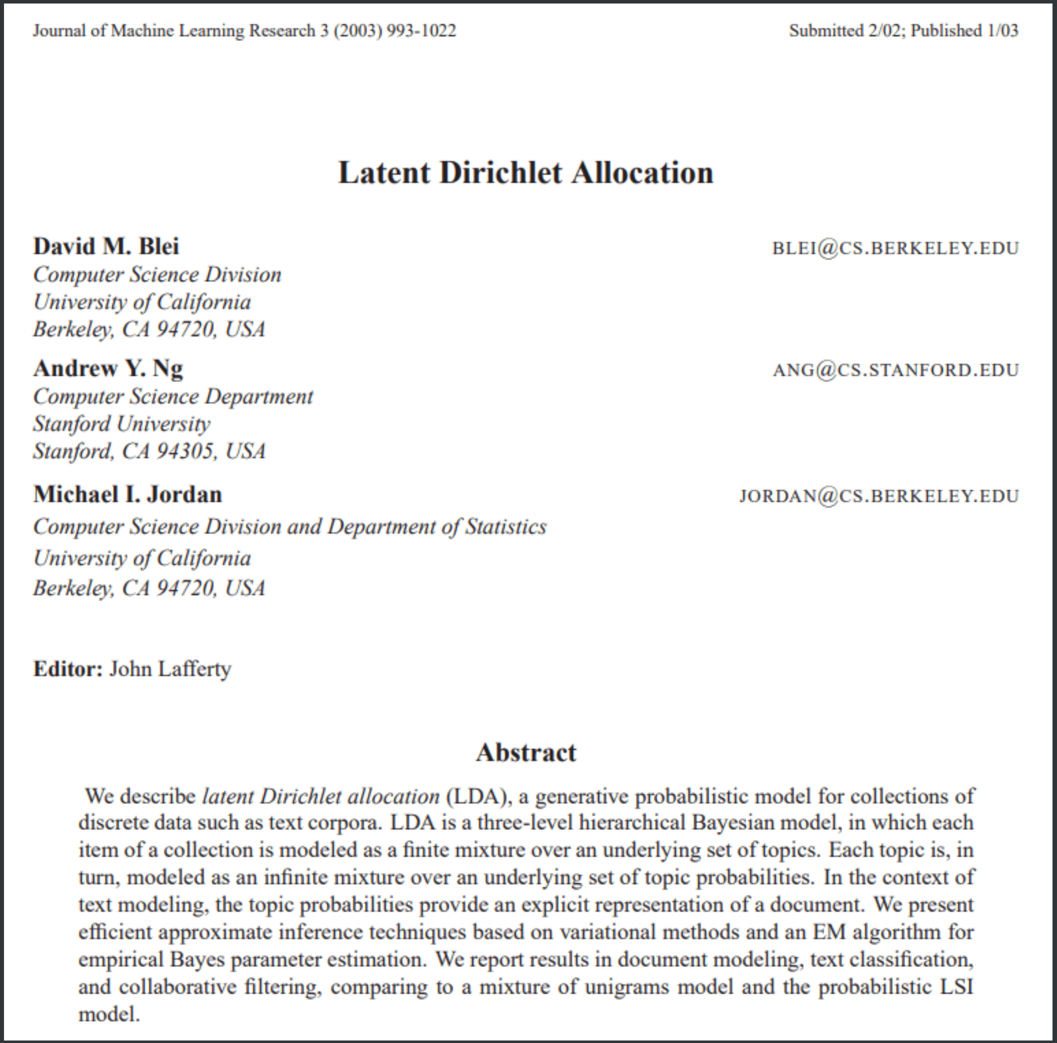
* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996

Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a famous topic modeling algorithm developed by three computer scientists, David Blei, Andrew Ng and Michael I. Jordan in 2003. Since then, LDA has become one of the most fundamental machine learning algorithms. You can find the original paper for LDA here.

You don’t need to understand all the mathematics. We just want to use the algorithm directly.



Journal of Machine Learning Research 3 (2003) 993-1022 Submitted 2/02; Published 1/03

Latent Dirichlet Allocation

David M. Blei BLEI@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720, USA

Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA

Editor: John Lafferty

Abstract

We describe *latent Dirichlet allocation* (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. We present efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation. We report results in document modeling, text classification, and collaborative filtering, comparing to a mixture of unigrams model and the probabilistic LSI model.

https://www.youtube.com/embed/p1I9Sa1lRvk?enablejsapi=1

Demonstration

Please visit here for an online demonstration of LDA.
(https://mimno.infosci.cornell.edu/jsLDA/jslda.html)

The source files are available on the course website.

You can also try the R code on the course website.

Additional Reading (optional):