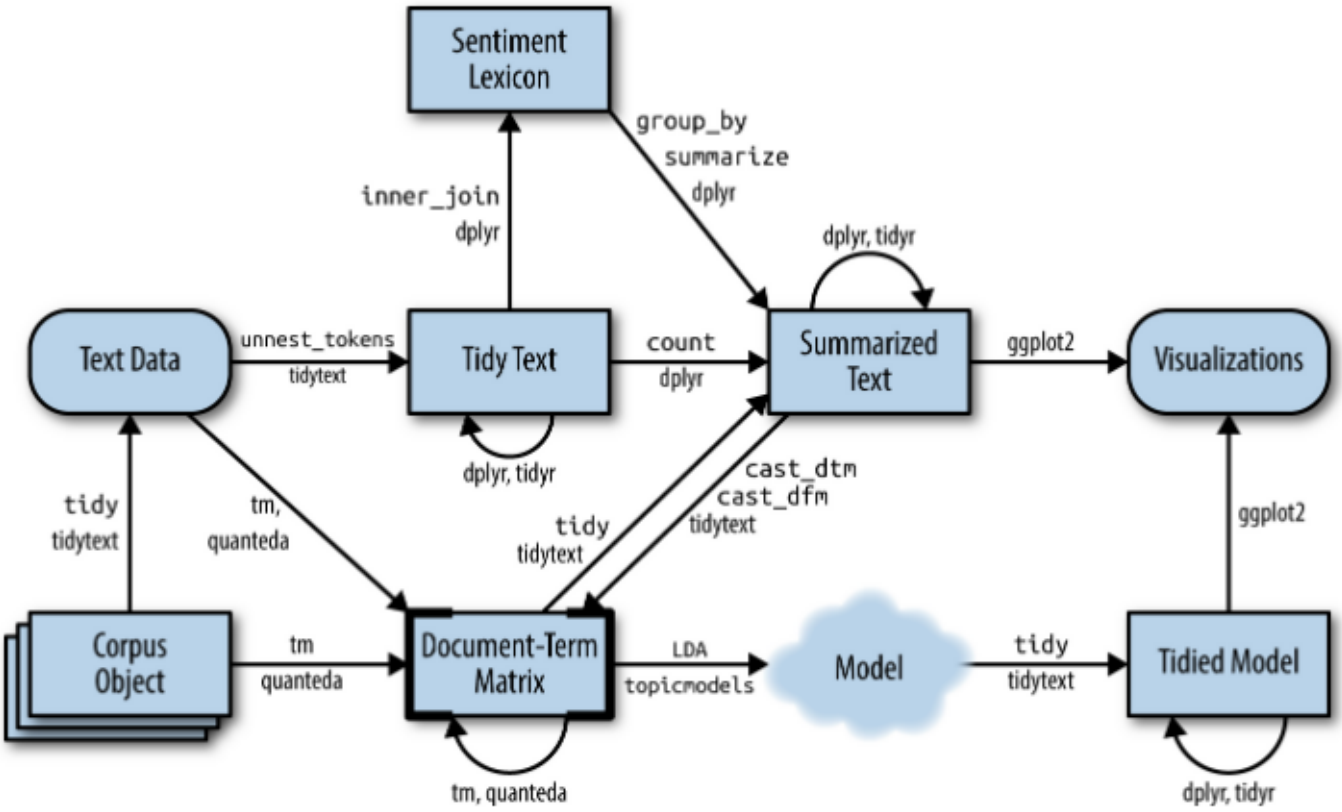
Chapter6. Topic modeling

Geonwoo Ban

텍스트 마이닝을 할 때면 종종 블로그 글들이나 뉴스 기사들과 같은 문서 모음집을 지니게 되는데, 이러한 문서 모음집을 구분 기준에 따라 그룹화를 함으로 꺼 각 그룹을 따로 이해할 수 있게 되기를 바랄 수 있다. **Topic modeling**은 무엇을 찾고자하는지를 잘 모를 때조차 숫자 데이터를 군집으로 처리해 그룹화를 하는 식으로 문서들을 비지도 방식으로 분류하는 방법이다.

Latent Dirichlet allocation(LDA)은 Topic model을 fitting하는 데 특히 많이 사용되는 방법이다. 이 방법에서는 여러 단어가 섞여 토픽을 이루고, 여러 토픽이 섞여 문서를 이룬다고 본다. 따라서 단어가 문장이 되고 문장이 문서가 되는 자연 언어의 전형적인 사용 방식을 반영해, 문서를 개별 그룹별로 분리하기보다는 각 문서의 내용이 서로 겹치게 할 수 있다.



위 사진에서 볼 수 있듯이 하나의 tidy structure tool로 topic model을 사용할 수 있다.

6.1 Latent Dirichlet allocation

NLP분야에서 LDA는 주어진 문서에 대하여 각 문서에 어떤 토픽들이 존재하는지를 서술하는 것에 대한 확률적 topic model 기법 중 하나이다. 알고 있는 주제별 단어수 분포를 바탕으로, 주어진 문서에서 발견된 단어수 분포를 분석함으로써 해당 문서가 어떤 주제들을 함께 다루고 있을지를 예측할 수 있다.

LDA는 topic modeling에 많이 사용되는 알고리즘 중 하나이고, 이 알고리즘에는 두 가지 원리가 있다.

- Every document is a mixture of topics.

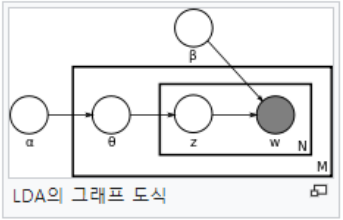
  각 문서에는 몇 가지 토픽에서 나온 단어가 특정 비율로 포함되어 있다고 생각한다. 예를 들어 토픽이 두 가지인 two-topic model에서 “문서1에서는 토픽 A가 90%를 차지하고 토픽 B가 10%를 차지하는 반면에, 문서2에서는 토픽 A가 30%를 차지하고 토픽 B가 70%를 차지한다”는 식으로 말 할 수 있다.

- Every topic is a mixture of words.

  예를 들어 ‘정치’에 관한 토픽과 ’연예’라는 토픽이 있는 두 가지 미국 뉴스를 생각해보면, 정치 토픽에서 가장 흔히 사용되는 단어는 ’대통령’, ‘의회’ 및 ‘정부’가 될 수 있지만 연예라는 토픽에 가장 흔히 사용되는 단어는 ’영화’, ‘텔레비젼’ 및 ‘배우’ 등 일 것이다. 하지만 토픽들이 같은 단어를 공유할 수도 있다는 점이 중요하다. 즉, 예를 들어 ’예산’과 같은 단어가 두 토픽에서 같이 나타날 수 있듯이 토픽이 달라도 같은 단어가 각 토픽에 공통으로 쓰일 수 있다.

LDA는 이 두 가지 경우를 동시에 추정하는 수학적 방법이다. LDA를 사용해서 각 토픽과 관련된 단어의 mixture가 무엇인지를 찾아낼 뿐만 아니라 각 문서를 설명하는 토픽의 mixture가 무엇인지를 결정한다.

모형 [편집]



LDA의 그래프 도식

M개의 문서가 주어져 있고, 모든 문서는 각각 k개의 주제 중 하나에 속할 때,

- 단어는 이산 자료의 기본 단위로 단어집(vocabulary)의 인덱스로 나타낼 수 있다. 단어집의 크기를 V라 하면, 각각의 단어는 인덱스 $v \in \{1, \dots, V\}$로 대응된다. 단어 벡터 w는 V-벡터로 표기하며 $w^v = 1, w^u = 0, u \neq v$를 만족한다.
- 문서는 N개의 단어의 연속으로 나타내며, $\mathbf{w} = (w_1, w_2, \dots, w_N)$ 으로 표기한다.
- 전집은 M개의 문서의 집합으로 나타내며, $D = \{\mathbf{w_1}, \mathbf{w_2}, \dots, \mathbf{w_M}\}$ 으로 표기한다.

LDA는 각각의 문서 $\mathbf{w} \in D$ 에 대해 다음과 같은 생성 과정을 가정한다.

1. $N \sim \text{Poisson}(\xi)$ 을 선택한다.

2. $\theta \sim \text{Dir}(\alpha)$ 를 선택한다.

3. 문서 내의 단어 $w_n \in \mathbf{w}$ 에 대해서

   (a) $z_n \sim \text{Multinomial}(\theta)$ 를 선택한다.

   (b) z_n이 주어졌을 때 w_n는 $p(w_n|z_n, \beta)$ 로부터 선택한다.

생성 과정에서 각각의 변수는 다음과 같은 의미를 가진다.

- α는 k 차원 디리클레 분포의 매개변수이다.
- θ는 k 차원 벡터이며, θ^i는 문서가 i번째 주제에 속할 확률 분포를 나타낸다.
- z_n는 k 차원 벡터이며, z_n^i는 단어 w_n이 i번째 주제에 속할 확률 분포를 나타낸다.
- β는 $k \times V$ 크기의 행렬 매개변수로, β_{ij}는 i번째 주제가 단어집의 j번째 단어를 생성할 확률을 나타낸다.

여기에서 w_n는 실제 문서를 통해 주어지며, 다른 변수는 관측할 수 없는 잠재 변수이다.

이 모형은 다음과 같이 해석될 수 있다. 각 문서에 대해 k개의 주제에 대한 가중치 θ가 존재한다. 문서 내의 각 단어 w_n은 k개의 주제에 대한 가중치 z_n을 가지는데, z_n은 θ에 의한 다항 분포로 선택된다. 마지막으로 실제 단어 w_n이 z_n에 기반하여 선택된다.

잠재 변수 α, β가 주어졌을 때 θ, $\mathbf{z} = \{z_1, \dots, z_N\}$, $\mathbf{w}$에 대한 결합 분포는 다음과 같이 구해진다.

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta),$$

여기서 z_n과 θ를 모두 더하여 문서의 주변 분포 (marginal distribution)를 구할 수 있다. 이 때 디 피네치의 정리 (de Finetti's theorem)에 의해 단어가 문서 안에서 교환성을 가지는 것을 확인할 수 있다.

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta$$

마지막으로 각각의 문서에 대한 주변 분포를 모두 곱하여 전집의 확률을 구할 수 있다.

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \right) d\theta_d$$

이 모형은 다음과 같이 해석될 수 있다. 각 문서에 대해 k개의 주제에 대한 가중치 θ 가 존재한다. 문서 내의 각 단어 w_n은 k개의 주제에 대한 가중치 z_n을 가지는데, z_n은 θ 에 의한 다항 분포로 선택된다. 마지막으로 실제 단어 w_n이 z_n에 기반하여 선택된다.

5장에서 DocumentTermMatrix의 예로서 topicmodels 패키지의 `AssociatedPress` 데이터셋을 간략하게 보았었다, 이것은 미국의 한 통신사에서 1988년경에 주로 작성한 2,246개 뉴스 기사를 모아 둔 Corpus이다.

```
library(topicmodels)

data("AssociatedPress")

AssociatedPress
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

LDA() 함수를 사용해 k=2로 설정하면 two-topic LDA model을 만들 수 있다.

```
# set a seed so that the output of the model is predictable
ap_lda <- LDA(AssociatedPress, k = 2, control = list(seed = 1234))
ap_lda
```

```
## A LDA_VEM topic model with 2 topics.
```

6.1.1 Word-topic probabilities

5장에서는 모델을 tidy structure로 바꿔주기 위해 `tidy()` 함수를 사용했다. tidytext 패키지는 LDA 모델을 통해 β라고 부르는 **per-topic-per-word probabilities**을 추출하는 방법을 제공한다.

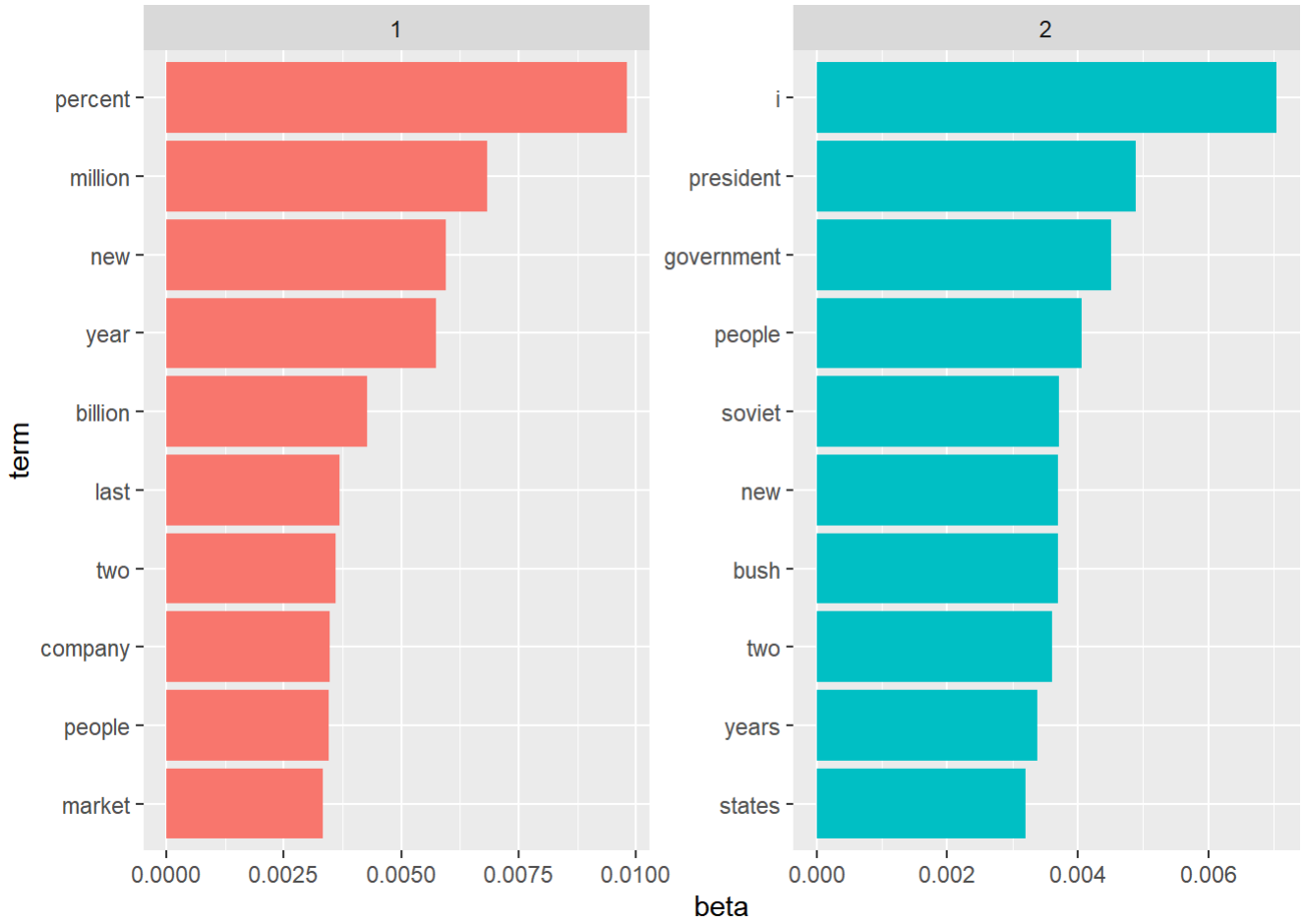
```
library(tidytext)

ap_topics <- tidy(ap_lda, matrix = "beta")
ap_topics
```

```
## # A tibble: 20,946 x 3
##    topic term           beta
##    <int> <chr>         <dbl>
##  1     1 aaron      1.69e-12
##  2     2 aaron      3.90e- 5
##  3     1 abandon    2.65e- 5
##  4     2 abandon    3.99e- 5
##  5     1 abandoned  1.39e- 4
##  6     2 abandoned  5.88e- 5
##  7     1 abandoning 2.45e-33
##  8     2 abandoning 2.34e- 5
##  9     1 abbott     2.13e- 6
## 10     2 abbott     2.97e- 5
## # ... with 20,936 more rows
```

이 코드가 모델을 one-topic-per-term-per-row 형식으로 바꾼 것을 주목해보면, 각 조합에 대해 모델은 해당 토픽에서 생성되는 용어의 확률을 계산한다.

dplyr의 `top_n()` 을 사용해 각 토픽에서 가장 흔한 용어 열 개를 찾을 수 있다. tidy data frame이기 때문에 ggplot2로 시각화하기 편하다.

```
library(ggplot2)
library(dplyr)

ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```



이 plot을 통해 기사에서 추출한 두 가지 토픽을 이해할 수 있다. 토픽 1에 가장 흔한 단어로 뽑힌 것들은 'percent', 'million', 'company' 등으로 경제/경영 또는 금융 뉴스를 나타낼 수 있음을 보인다. 토픽 2에서 가장 흔하게 사용되는 용어는 'president', 'goverment', 'soviet' 등으로 이 토픽은 정치 뉴스를 나타낸다. 각 토픽의 단어에 대한 중요한 관찰 중 하나는 'new' 및 'people'과 같은 일부 단어가 두 토픽에 공통적으로 나타난다는 것이다. 이것은 **hard clustering** 방법과 반대되는 토픽 모델링의 장점이다.(<-> soft clustering)

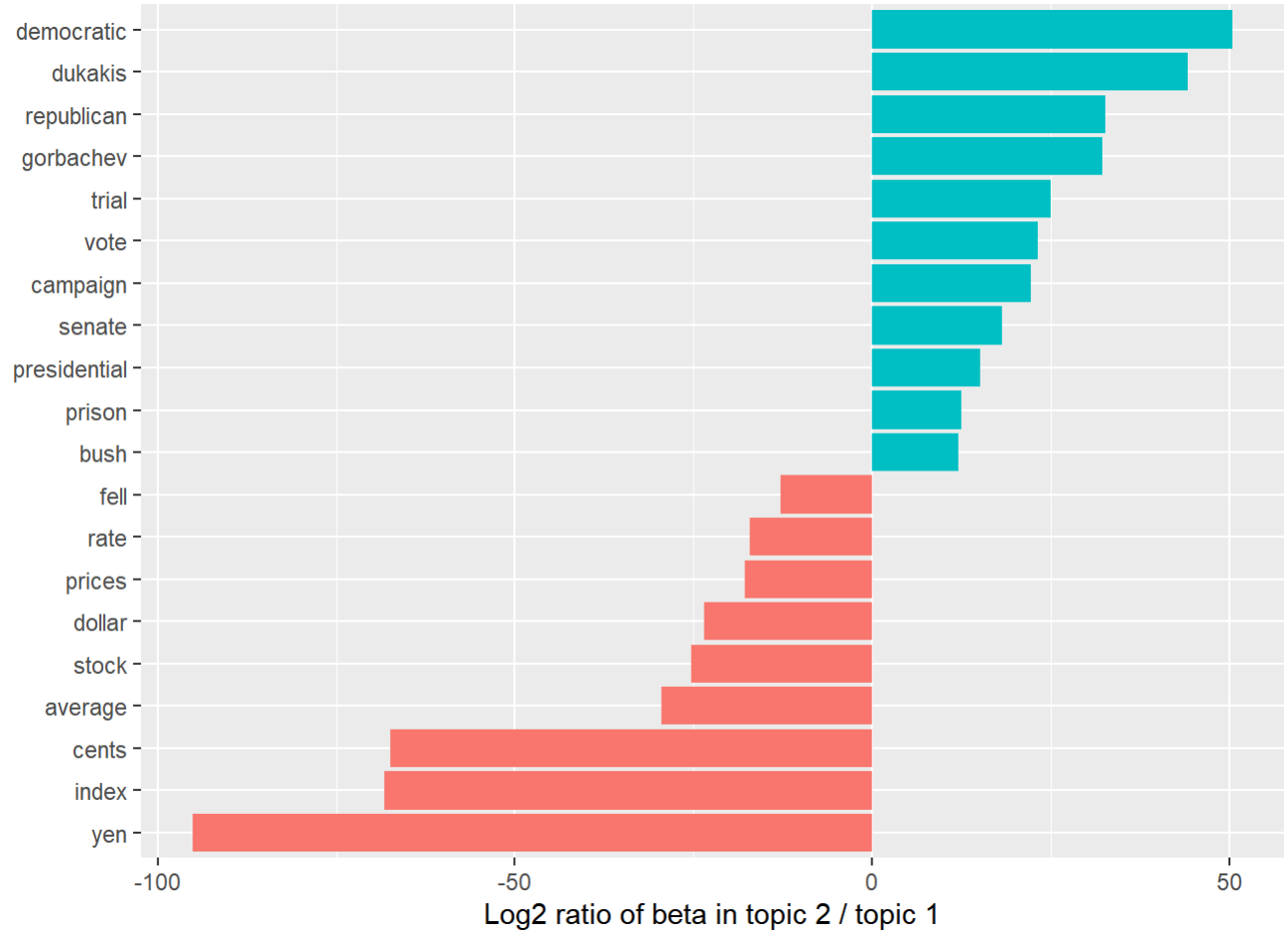
공통적인 단어들이 주 목적이 아닌 토픽간 확실한 차이를 보기위해서 토픽1과 토픽2 사이의 β 값의 차이가 가장 큰 용어들을 고려할 수 있다. 이는 두 로그 비율을 기반으로 추정할 수 있다: $log_2(\frac{\beta_2}{\beta_1})$, β_2가 두 배 더 크면 로그비율은 1이 되고 β_1이 두 배 더 크면 -1이 된다.

특히 관련성 높은 단어 집합으로 제한하기 위해 적어도 하나의 토픽에서 1/1000보다 큰 단어와 같이 상대적으로 흔한 단어를 선별할 수 있다.

```
library(tidyr)

beta_wide <- ap_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  pivot_wider(names_from = topic, values_from = beta) %>% # topic별로 beta값을 보기위해 wide form으로 변환.
  filter(topic1 > .001 | topic2 > .001) %>% # 각 topic에서 beta값이 0.001보다 큰 단어들만 선별.
  mutate(log_ratio = log2(topic2 / topic1)) # 로그비 계산.

beta_wide
```

```
## # A tibble: 198 x 4
##    term              topic1      topic2 log_ratio
##    <chr>              <dbl>       <dbl>     <dbl>
##  1 administration 0.000431   0.00138         1.68
##  2 ago            0.00107    0.000842       -0.339
##  3 agreement      0.000671   0.00104         0.630
##  4 aid            0.0000476 0.00105         4.46
##  5 air            0.00214    0.000297       -2.85
##  6 american       0.00203    0.00168        -0.270
##  7 analysts       0.00109    0.000000578   -10.9
##  8 area           0.00137    0.000231       -2.57
##  9 army           0.000262   0.00105         2.00
## 10 asked          0.000189   0.00156         3.05
## # ... with 188 more rows
```

```
beta_wide %>%
  top_n(n = 20,abs(log_ratio)) %>%
  mutate(term=reorder(term,log_ratio)) %>%
  ggplot(aes(x=log_ratio, y=term, fill=log_ratio>0))+geom_bar(stat='identity',show.legend=F)+
  labs(x="Log2 ratio of beta in topic 2 / topic 1",y="")
```



두 토픽 사이의 가장 큰 차이점을 가진 단어에 대한 그림이며, 이 그림을 통해 토픽 2에서 많이 보이는 단어가 'democratic'과 'republican'과 같은 정당과 'dukakis'와 'gorbachev'와 같은 정치인의 이름을 포함하고 있음을 볼 수 있다. 토픽 1은 'yen' 및 'dollar'와 같은 통화뿐만 아니라 'index', 'prices', 'rate'와 같은 재정적 용어들을 특징으로 한다. 이는 알고리즘이 식별한 두 토픽이 정치 뉴스와 금융 뉴스라는 점을 확인하는 데 도움이 된다.

6.1.2 Document-topic probabilities

각 토픽을 단어의 mixture라고 추정하는 일 외에도 LDA는 각 문서를 토픽의 mixture인 것으로 보고 모델링을 한다. `tidy()` 에서 `matrix="gamma"` 라는 옵션을 사용해 γ라는 **per-document-per-topic probabilities**을 계산할 수 있다.

```
ap_documents <- tidy(ap_lda, matrix = "gamma")
ap_documents
```

```
## # A tibble: 4,492 x 3
##    document topic    gamma
##       <int> <int>    <dbl>
##  1        1     1 0.248
##  2        2     1 0.362
##  3        3     1 0.527
##  4        4     1 0.357
##  5        5     1 0.181
##  6        6     1 0.000588
##  7        7     1 0.773
##  8        8     1 0.00445
##  9        9     1 0.967
## 10       10     1 0.147
## # ... with 4,482 more rows
```

이러한 값들은 각 해당 토픽으로부터 생성된 해당 문서의 단어 추정 비율이다. 예를 들어 문서 1의 단어 중 약 24.8%만 토픽 1에서 생성됨을 해석할 수 있다.

대부분의 문서들은 두 토픽을 골고루 혼합하여 작성된 것을 알 수 있지만, 문서6의 경우엔 대부분의 단어가 토픽 2에서 도출되었으며, 토픽 1에서 나온 γ는 0에 가깝다. 해당 문서에서 가장 흔한 단어가 무엇인지를 확인해 볼 수 있다.

```
tidy(AssociatedPress) %>%
  filter(document == 6) %>%
  arrange(desc(count))
```

```
## # A tibble: 287 x 3
##    document term           count
##       <int> <chr>          <dbl>
##  1        6 noriega           16
##  2        6 panama            12
##  3        6 jackson            6
##  4        6 powell             6
##  5        6 administration     5
##  6        6 economic           5
##  7        6 general            5
##  8        6 i                  5
##  9        6 panamanian         5
## 10        6 american           4
## # ... with 277 more rows
```

가장 흔한 단어를 바탕으로 추측하건대 이 문서는 미국 정부와 파나마 독재자 “Manuel Noriega” 사이의 관계에 관한 기사인 것으로 보인다. 이는 알고리즘이 기사를 토픽 2(정치/국가)에 배치하는 것이 옳았다는 것을 의미한다.

6.2 Example: the great library heist

비지도학습의 경우 분류 모델에 대한 정답을 알 수 없어서 모델을 평가하는데 어려움이 있다. 이번에는 네 개 개별 토픽과 관련이 있는 문서들을 수집한 다음 토픽 모델링을 수행하여 알고리즘이 네 개 그룹을 정확하게 구별할 수 있는지 여부를 확인할 수 있다. 이를 통해 이 방법이 유용하다는 점을 확인할 수 있고, 언제 어떻게 잘못될 수 있는지에 대한 정보도 얻을 수 있다.

만약 어떤 사람이 네 권의 책을 찢어버려 각 장별로 분해된 상태에서 큰 파일에 담겨있게 되었다고 하면, 어떻게 해야 이렇게 뒤섞인 장들을 원래의 도서로 복원할 수 있을까?

- *Great Expectations*(위대한 유산) by Charles Dickens
- *The War of the Worlds*(우주 전쟁) by H.G. Wells
- *Twenty Thousand Leagues Under the Sea*(해저 2만리) by Jules Verne
- *Pride and Prejudice*(오만과 편견) by Jane Austen

이런 문제는 각 장에 레이블이 지정되어 있지 않으므로 까다롭다. 각 장을 어떤 단어들을 기준으로 삼아 한군데로 모을 수 있을지 모르기 때문이다. 따라서 토픽 모델링을 사용해 각 장이 어떻게 개별 토픽으로 군집화되는지를 알아낼 생각이다. 하나의 가설로 각 장은 도서들 중 한 개를 대표할 것이다.

```
titles <- c("Twenty Thousand Leagues under the Sea",
            "The War of the Worlds",
            "Great Expectations")

library(gutenbergr)

books <- gutenberg_works(title %in% titles) %>%
  gutenberg_download(meta_fields = "title")
books
```

```
## # A tibble: 39,006 x 3
##    gutenberg_id text                                                title
##           <int> <chr>                                               <chr>
##  1           36 "The War of the Worlds"                             The War of the ~
##  2           36 ""                                                  The War of the ~
##  3           36 "by H. G. Wells [1898]"                             The War of the ~
##  4           36 ""                                                  The War of the ~
##  5           36 ""                                                  The War of the ~
##  6           36 "     But who shall dwell in these worlds if t~     The War of the ~
##  7           36 "     inhabited? .  .  .  Are we or they Lords~     The War of the ~
##  8           36 "     World? .  .  .  And how are all things m~     The War of the ~
##  9           36 "          KEPLER (quoted in The Anatomy of Me~     The War of the ~
## 10           36 ""                                                  The War of the ~
## # ... with 38,996 more rows
```

전처리 과정에서 이들을 장별로 나누고 tidytext의 `unnest_tokens()` 를 사용해 단어로 분리한 다음 stop_words를 제거한다. 여기서 분리한 모든 장을 별도의 문서로 취급한다. 각 장은 Great_Expectations_1 또는 Pride and Prejudice_11과 같은 이름으로 되어 있다.

```
library(stringr)
library(janeaustenr)

austen_books() %>%
  group_by(book) %>%
  mutate(chapter = cumsum(str_detect(
    text,regex("^chapter ",ignore_case=TRUE)))) %>%
  ungroup() %>%
  filter(book=="Pride & Prejudice") %>%
  filter(chapter > 0) %>%
  unite(document, book, chapter) -> Pride


# divide into documents, each representing one chapter
by_chapter <- books %>%
  group_by(title) %>%
  mutate(chapter = cumsum(str_detect(
    text, regex("^chapter ", ignore_case = TRUE)
  ))) %>%
  ungroup() %>%
  filter(chapter > 0) %>%
  unite(document, title, chapter) %>%
  select(text, document)

by_chapter <- rbind(Pride, by_chapter)
by_chapter
```

```
## # A tibble: 51,909 x 2
##    text                                                     document
##    <chr>                                                    <chr>
##  1 "Chapter 1"                                              Pride & Prejudi~
##  2 ""                                                       Pride & Prejudi~
##  3 ""                                                       Pride & Prejudi~
##  4 "It is a truth universally acknowledged, that a single man ~ Pride & Prejudi~
##  5 "of a good fortune, must be in want of a wife."          Pride & Prejudi~
##  6 ""                                                       Pride & Prejudi~
##  7 "However little known the feelings or views of such a man m~ Pride & Prejudi~
##  8 "first entering a neighbourhood, this truth is so well fixe~ Pride & Prejudi~
##  9 "of the surrounding families, that he is considered the rig~ Pride & Prejudi~
## 10 "of some one or other of their daughters."               Pride & Prejudi~
## # ... with 51,899 more rows
```

```
# split into words
by_chapter_word <- by_chapter %>%
  unnest_tokens(word, text)
by_chapter_word
```

```
## # A tibble: 473,016 x 2
##    document            word
##    <chr>               <chr>
##  1 Pride & Prejudice_1 chapter
##  2 Pride & Prejudice_1 1
##  3 Pride & Prejudice_1 it
##  4 Pride & Prejudice_1 is
##  5 Pride & Prejudice_1 a
##  6 Pride & Prejudice_1 truth
##  7 Pride & Prejudice_1 universally
##  8 Pride & Prejudice_1 acknowledged
##  9 Pride & Prejudice_1 that
## 10 Pride & Prejudice_1 a
## # ... with 473,006 more rows
```

```
# find document-word counts
word_counts <- by_chapter_word %>%
  anti_join(stop_words) %>%
  count(document, word, sort = TRUE) %>%
  ungroup()
word_counts
```

```
## # A tibble: 105,548 x 3
##    document                 word        n
##    <chr>                    <chr>   <int>
##  1 Great Expectations_57    joe        88
##  2 Great Expectations_7     joe        70
##  3 Great Expectations_17    biddy      63
##  4 Great Expectations_27    joe        58
##  5 Great Expectations_38    estella    58
##  6 Great Expectations_2     joe        56
##  7 Great Expectations_23    pocket     53
##  8 Great Expectations_15    joe        50
##  9 Great Expectations_18    joe        50
## 10 The War of the Worlds_16 brother    50
## # ... with 105,538 more rows
```

6.2.1 LDA on chapters

앞서 텍스트 데이터들을 `tidy structure` 로 만들었지만, topicmodel을 만들기 위해서는 DTM형식의 데이터가 필요하다. tidytext 패키지의 `cast_dtm()` 을 사용하여 DTM형식의 데이터로 변환할 수 있다.

```
chapters_dtm <- word_counts %>%
  cast_dtm(document, word, n)
chapters_dtm
```

```
## <<DocumentTermMatrix (documents: 193, terms: 18316)>>
## Non-/sparse entries: 105548/3429440
## Sparsity           : 97%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

그런 다음 `LDA()` 함수를 사용하여 `k=4` 인 토픽모델을 만들 수 있다.

```
chapters_lda <- LDA(chapters_dtm, k = 4, control = list(seed = 1234))
chapters_lda
```

```
## A LDA_VEM topic model with 4 topics.
```

다음으로 각 단어 별로 각 토픽에 속할 확률을 조사할 수 있다.

```
chapter_topics <- tidy(chapters_lda, matrix = "beta")
chapter_topics
```

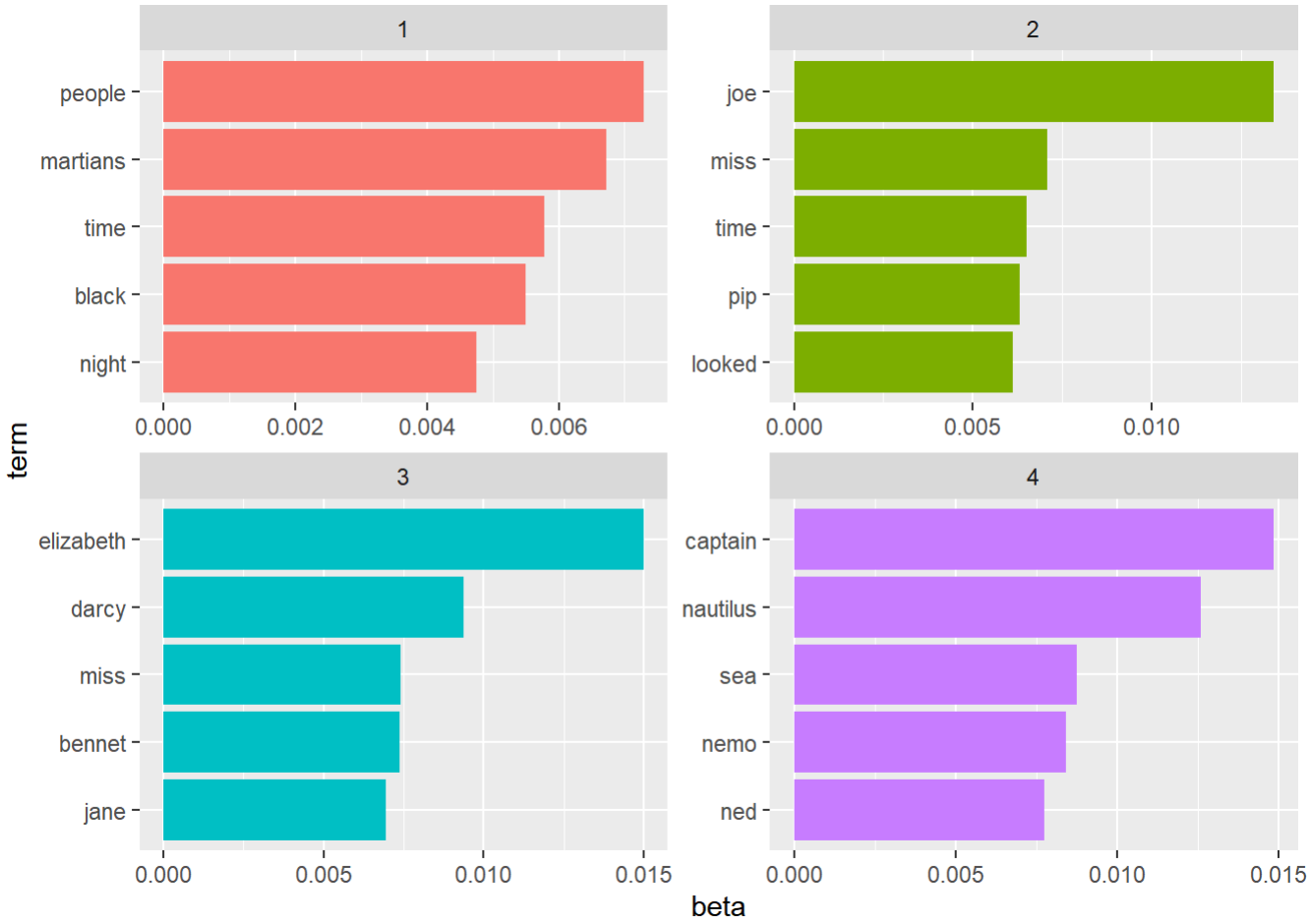
```
## # A tibble: 73,264 x 3
##    topic term         beta
##    <int> <chr>       <dbl>
##  1     1 joe     2.61e- 30
##  2     2 joe     1.34e-  2
##  3     3 joe     5.11e- 70
##  4     4 joe     2.28e- 89
##  5     1 biddy   5.06e- 37
##  6     2 biddy   4.42e-  3
##  7     3 biddy   4.72e- 53
##  8     4 biddy   1.67e-102
##  9     1 estella 7.89e-  4
## 10     2 estella 4.23e-  3
## # ... with 73,254 more rows
```

이를 사용하여 각 토픽 내에서 상위 다섯 개 용어를 찾아보자.

```
top_terms <- chapter_topics %>%
  group_by(topic) %>%
  top_n(5, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
top_terms
```

```
## # A tibble: 20 x 3
##    topic term         beta
##    <int> <chr>       <dbl>
##  1     1 people    0.00729
##  2     1 martians  0.00672
##  3     1 time      0.00578
##  4     1 black     0.00550
##  5     1 night     0.00476
##  6     2 joe       0.0134
##  7     2 miss      0.00709
##  8     2 time      0.00651
##  9     2 pip       0.00632
## 10     2 looked    0.00611
## 11     3 elizabeth 0.0150
## 12     3 darcy     0.00937
## 13     3 miss      0.00740
## 14     3 bennet    0.00739
## 15     3 jane      0.00694
## 16     4 captain   0.0149
## 17     4 nautilus  0.0126
## 18     4 sea       0.00877
## 19     4 nemo      0.00842
## 20     4 ned       0.00776
```

```
library(ggplot2)
top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```



- Topic1 : martians, black, night -> The war of the world
- Topic2 : joe, pip -> Great Expectations
- Topic3 : elizabeth, darcy -> Pride & Prejudice
- Topic4 : captain, nautilus, sea -> Twenty Thousand Leagues Under the Sea
- LDA가 **fuzzy clustering** 방법에 따라, miss, time과 같은 단어처럼 여러 토픽 사이에 공통된 단어가 있을 수 있음을 볼 수 있다.
    - **fuzzy clustering** : Soft clustering or Soft k-means

6.2.2 Per-document classification

현재 분석에서는 1개의 Chapter를 하나의 문서로 생각하고 진행한다. 우리는 어떤 토픽이 각 문서와 관련되어 있는지를 알고 싶을 때, 토픽 별 문서에 대한 확률인 γ를 통해 이를 확인할 수 있다.

```
chapters_gamma <- tidy(chapters_lda, matrix = "gamma")
chapters_gamma
```

```
## # A tibble: 772 x 3
##    document                topic     gamma
##    <chr>                   <int>     <dbl>
##  1 Great Expectations_57       1 0.0000137
##  2 Great Expectations_7        1 0.0000145
##  3 Great Expectations_17       1 0.0000212
##  4 Great Expectations_27       1 0.0000193
##  5 Great Expectations_38       1 0.279
##  6 Great Expectations_2        1 0.0000171
##  7 Great Expectations_23       1 0.0000188
##  8 Great Expectations_15       1 0.0000144
##  9 Great Expectations_18       1 0.0000128
## 10 The War of the Worlds_16    1 0.740
## # ... with 762 more rows
```

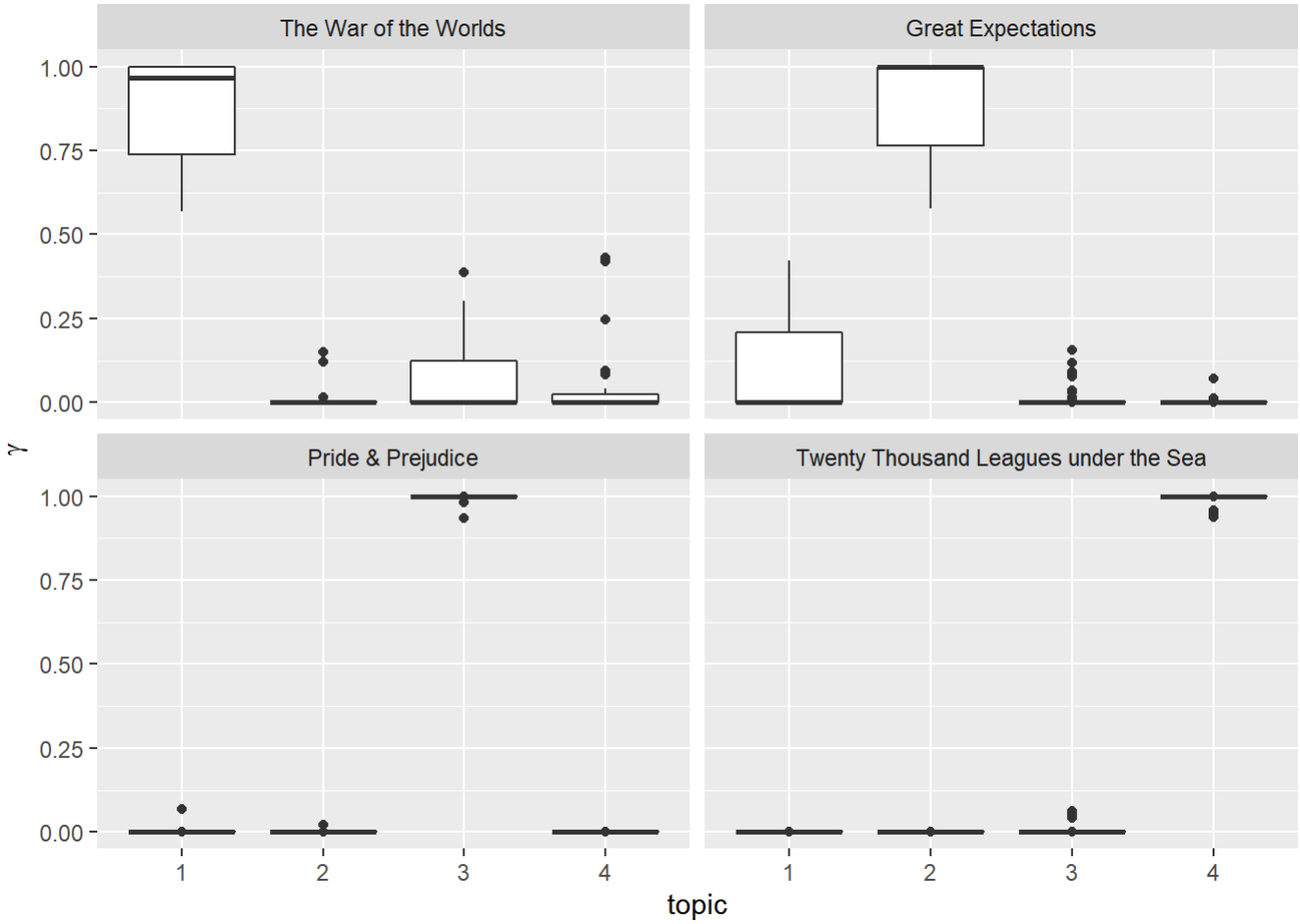
이러한 값들은 각기 해당 토픽으로부터 생성된 단어들의 각 문서에 대한 추정 비율이다. 예를 들어 Great Expectations_57 문서의 각 단어가 토픽 1에서 나올 확률이 0.000135%라는 것을 의미한다.

이러한 확률을 통해 비지도 학습이 네 권의 도서를 구별하는 데 얼마나 효과적이었는지 알 수 있다. 이를 확인하기 위해 제목과 장으로 문서 이름을 다시 분리한 다음, 각 도서 내의 각 장에 대한 감마확률의 Box plot을 확인해보자.

```
chapters_gamma <- chapters_gamma %>%
  separate(document, c("title", "chapter"), sep = "_", convert = TRUE)

chapters_gamma
```

```
## # A tibble: 772 x 4
##    title                chapter topic     gamma
##    <chr>                  <int> <int>     <dbl>
##  1 Great Expectations        57     1 0.0000137
##  2 Great Expectations         7     1 0.0000145
##  3 Great Expectations        17     1 0.0000212
##  4 Great Expectations        27     1 0.0000193
##  5 Great Expectations        38     1 0.279
##  6 Great Expectations         2     1 0.0000171
##  7 Great Expectations        23     1 0.0000188
##  8 Great Expectations        15     1 0.0000144
##  9 Great Expectations        18     1 0.0000128
## 10 The War of the Worlds     16     1 0.740
## # ... with 762 more rows
```

```
chapters_gamma %>%
  mutate(title = reorder(title, gamma * topic)) %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~ title) +
  labs(x = "topic", y = expression(gamma))
```



각 도서에 대해 문서별 Topic에 대한 확률의 분포를 보면, 대부분 각기 하나의 토픽이 각 도서에 할당되어 도서간의 식별이 잘 되어있음을 알 수 있다.

과연 LDA가 각 Chapter를 도서별로 잘 정리를 하였는가를 알아보기 위해 각 Chapter별 가장 높은 γ값을 가지는 Topic을 출력하여 각 Chapter와 가장 관련이 있는 Topic을 찾을 수 있다.

```
chapter_classifications <- chapters_gamma %>%
  group_by(title, chapter) %>%
  slice_max(gamma) %>%
  ungroup()

chapter_classifications
```

```
## # A tibble: 193 x 4
##    title             chapter topic gamma
##    <chr>               <int> <int> <dbl>
##  1 Great Expectations      1     2 1.00
##  2 Great Expectations      2     2 1.00
##  3 Great Expectations      3     2 1.00
##  4 Great Expectations      4     2 1.00
##  5 Great Expectations      5     2 1.00
##  6 Great Expectations      6     2 1.00
##  7 Great Expectations      7     2 1.00
##  8 Great Expectations      8     2 0.648
##  9 Great Expectations      9     2 1.00
## 10 Great Expectations     10     2 1.00
## # ... with 183 more rows
```

그런 다음 같은 title 중 다른 topic을 가지는 Chapter를 찾을 수 있다.

```
book_topics <- chapter_classifications %>%
  count(title, topic) %>%
  group_by(title) %>%
  slice_max(n, n = 1) %>%
  ungroup() %>%
  transmute(consensus = title, topic)

chapter_classifications %>%
  inner_join(book_topics, by = "topic") %>%
  filter(title != consensus)
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: title <chr>, chapter <int>, topic <int>, gamma <dbl>,
## #   consensus <chr>
```

본 경우에서는 섞여있던 Chapter들을 모두 정확하게 원래의 책끼리 Grouping이 됨을 볼 수 있습니다.

6.2.3 By word assignments: augment

LDA 알고리즘 중 한 단계는 각 문서의 각 단어를 토픽에 할당하는 것이다. 문서 내 더 많은 단어가 해당 토픽에 할당되면 일반적으로 더 많은 가중치(gamma)가 해당 문서-토픽 분류에 부여된다.

원본 문서-단어 쌍을 가져와서 각 문서에서 어떤 단어가 어떤 토픽에 할당되었는지 찾아야 할 때가 있다. 이 경우 broom 패키지의 `augment()` 함수를 통해 확인할 수 있다.

```
assignments <- augment(chapters_lda, data = chapters_dtm)
assignments
```

```
## # A tibble: 105,548 x 4
##    document              term  count .topic
##    <chr>                 <chr> <dbl>  <dbl>
##  1 Great Expectations_57 joe      88      2
##  2 Great Expectations_7  joe      70      2
##  3 Great Expectations_17 joe       5      2
##  4 Great Expectations_27 joe      58      2
##  5 Great Expectations_2  joe      56      2
##  6 Great Expectations_23 joe       1      2
##  7 Great Expectations_15 joe      50      2
##  8 Great Expectations_18 joe      50      2
##  9 Great Expectations_9  joe      44      2
## 10 Great Expectations_13 joe      40      2
## # ... with 105,538 more rows
```

이렇게 하면 도서-용어 빈도수들로 구성된 tidy data structure가 반환되지만, 각 용어가 각 문서 내에서 할당된 토픽과 함께 추가 열인 .topic이 추가된다.

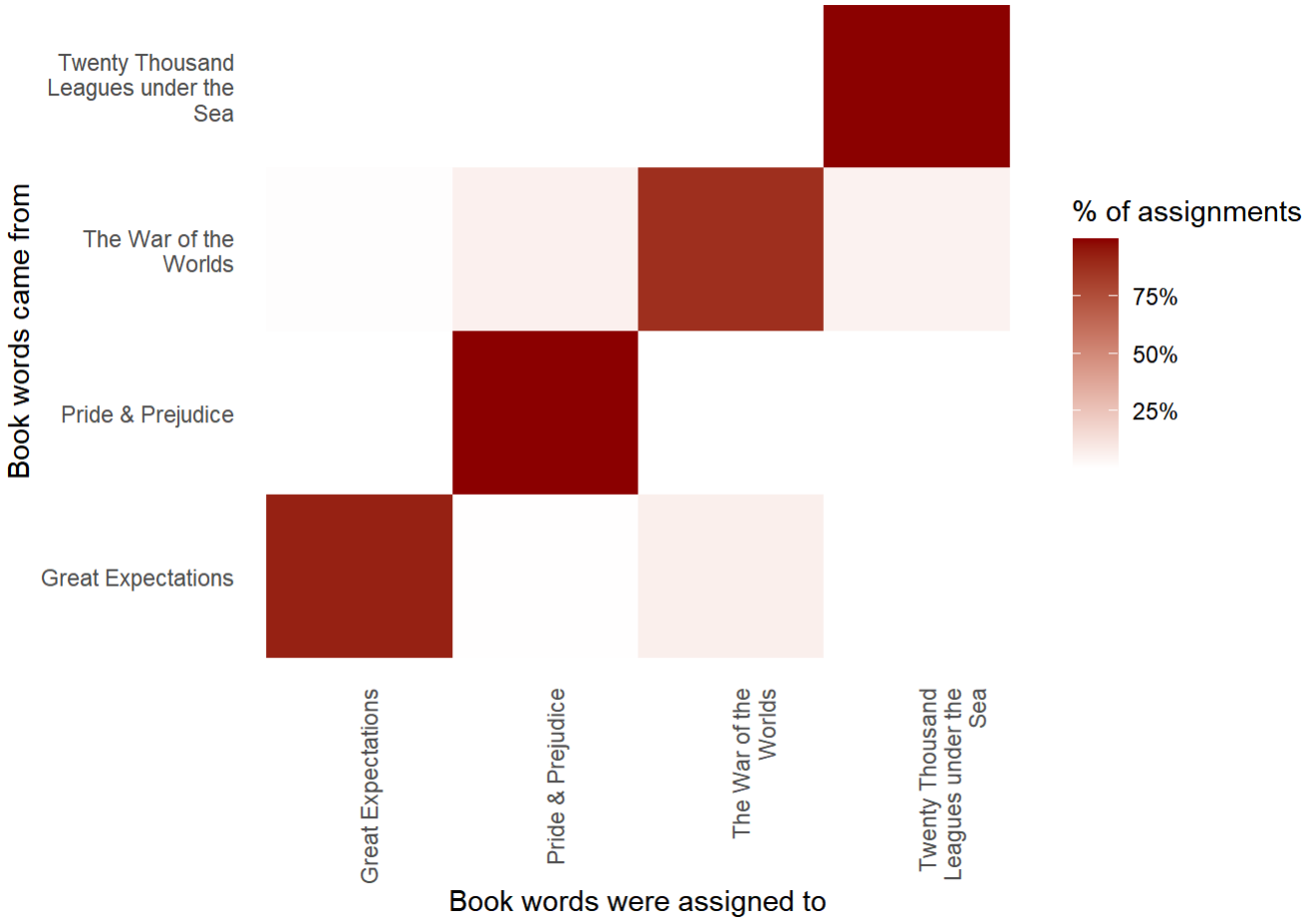
이를 통해 어떤 단어들이 다른 Topic으로 잘못 분류되었는지를 찾을 수 있다.

```
assignments <- assignments %>%
  separate(document, c("title", "chapter"),
           sep = "_", convert = TRUE) %>%
  inner_join(book_topics, by = c(".topic" = "topic"))

assignments
```

```
## # A tibble: 105,548 x 6
##    title              chapter term  count .topic consensus
##    <chr>                <int> <chr> <dbl>  <dbl> <chr>
##  1 Great Expectations      57 joe      88      2 Great Expectations
##  2 Great Expectations       7 joe      70      2 Great Expectations
##  3 Great Expectations      17 joe       5      2 Great Expectations
##  4 Great Expectations      27 joe      58      2 Great Expectations
##  5 Great Expectations       2 joe      56      2 Great Expectations
##  6 Great Expectations      23 joe       1      2 Great Expectations
##  7 Great Expectations      15 joe      50      2 Great Expectations
##  8 Great Expectations      18 joe      50      2 Great Expectations
##  9 Great Expectations       9 joe      44      2 Great Expectations
## 10 Great Expectations      13 joe      40      2 Great Expectations
## # ... with 105,538 more rows
```

Real title과 각 Chapter에 할당된 hat title로 이루어진 데이터셋을 가지고 혼동행렬을 시각화할 수 있다.

```
library(scales)

assignments %>%
  count(title, consensus, wt = count) %>%
  mutate(across(c(title, consensus), ~str_wrap(., 20))) %>%
  group_by(title) %>%
  mutate(percent = n / sum(n)) %>%
  ggplot(aes(consensus, title, fill = percent)) +
  geom_tile() +
  scale_fill_gradient2(high = "darkred", label = percent_format()) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        panel.grid = element_blank()) +
  labs(x = "Book words were assigned to",
       y = "Book words came from",
       fill = "% of assignments")
```


Twenty Thousand Leagues under the Sea
The War of the Worlds
Pride & Prejudice
Great Expectations
Book words came from
Great Expectations
Pride & Prejudice
The War of the Worlds
Twenty Thousand Leagues under the Sea
Book words were assigned to
% of assignments
75%
50%
25%

Summary

이번 장에서는 하나의 문서를 특징을 잡는 단어 군집을 찾기 위한 토픽 모델링을 소개하고, dplyr 및 ggplot2를 사용하여 모델을 탐색하고 이해하는 방법을 보여주었다.

다양한 출력 형식의 문제를 tidy function들이 처리하고, tool을 사용하여 모델 결과를 탐색할 수 있다는 장점이 tidy approach의 장점 중 하나이다.

토픽 모델링을 통해 네 개의 개별 도서와 장을 구분하고 구별할 수 있다는 점을 보았고, 잘못 지정된 단어와 장을 찾아보며 모델의 한계점 또한 볼 수 있었다.

Measuring Clustering Quality

https://www.sciencedirect.com/topics/computer-science/clustering-quality
(https://www.sciencedirect.com/topics/computer-science/clustering-quality)

어떤 방법에 의해 생성된 클러스터링은 얼마나 좋은 것인가에 대한 Clustering Quality를 측정하는 방법은 실제 cluster의 값을 알 수 있을 때와 알수 없을 때로 나누어 계산을 하게 된다.

> Four essential criteria

Clustering Quality에 대한 측정은 일반적으로 다음과 같은 4가지 필수 기준을 만족하는 경우에 효과적이다.

- Cluster homogeneity : 나어진 cluster 내의 동질성이 높으면 quality에 더 높은 점수를 주어야함. Pure해야함.
- Cluster completeness : 개체가 동일한 범주라면 동일한 cluster에 속하여야 한다.
- Rag bag : 다른 개체와 병합할 수 없는 개체들을 Rag bag이라고 한다. 이러한 개체들을 각각 다른 cluster에 넣기 보다는 한번에 모아 Rag bag이라는 cluster로 분류해야한다.
- Small cluster preservation : 큰 cluster를 나누는 것보다 작은 cluster를 나누는 것이 더 안좋은 방법이다.
    - ex) A : (a1,a2,a3,a4), B : (b1,b2)라는 cluster가 있고 만약 cluster 개수를 늘리고 싶을 때 B를 쪼개기 보다는 A를 쪼개는 것이 더 좋은 방법이다.

BCubed

위 네 가지 기준을 모두 충족하는 측도 중 하나는 **BCubed precision and recall**이다. 이 측도는 실제 cluster를 알 수 있을 때 사용할 수 있다(extrinsic methods).

+Precision은 동일한 클러스터에 있는 다른 객체가 객체와 동일한 범주에 속하는 개수를 반영한다.

+Recall은 동일한 범주의 개체가 동일한 클러스터에 할당되는 개수를 반영한다.

silhouette coefficient

silhouette coefficient는 실제 cluster을 알 수 없을 때 사용하는 측도이다(instrinsic methods). 이러한 경우에는 군집들이 얼마나 잘 분리되어 있고, 군집들이 얼마나 작은지 검토하여 평가를 하게 된다.

- silhouette coefficient를 계산하기 위해, 먼저 n개의 관측치 데이터셋을 D라고 했을 때, D는 k개의 cluster로 분할이 된다고 가정해보자.
- 각 관측치 o는 D에 속해있고, 각 관측치 o에 대해 o가 속한 cluster의 다른 모든 개체 사이의 평균 거리 a(o)를 계산한다.
- 마찬가지로 o에서 o가 속하지 않는 모든 군집까지의 최소 평균 거리 b(o)를 계산하여 a(o)와 b(o)를 사용하여 sihouette coefficient를 계산하게 된다.

$$s(o) = \frac{b(o)-a(o)}{max\{a(o),b(o)\}}$$ </ center>

- a(o)는 cluster의 compactness를 반영하며, 값이 작을수록 cluster가 더 압축되었다는 것을 의미한다.
- b(o)는 o가 다른 군집과 분리되어있는 정도를 반영하며, 값이 클수록 o는 다른 군집과 더 많이 분리가 되어있다는 것을 의미한다.
- 따라서 o의 silhouette이 1에 가까워지면 o를 포함하는 cluster는 compact하고 o는 다른 군집과 멀리 떨어져 있는 것을 의미하게 된다.
- 그러나 s(o)가 음수인 경우, 이는 o가 o와 같은 군집의 개체보다 다른 군집의 개체들에 더 가깝다는 것을 의미한다. 이 경우는 clustering이 잘 되지 못하였다는 것을 의미하게 된다.
- cluster 내에서 cluster의 적합성을 측정하기 위해 cluster 내 모든 개체의 평균 s(o)를 계산할 수 있다.
- clustering의 quality를 측정하기 위해 data set에 있는 모든 객체의 평균 s(o)를 사용한다.