Chapter5. Converting to and from non-tidy formats

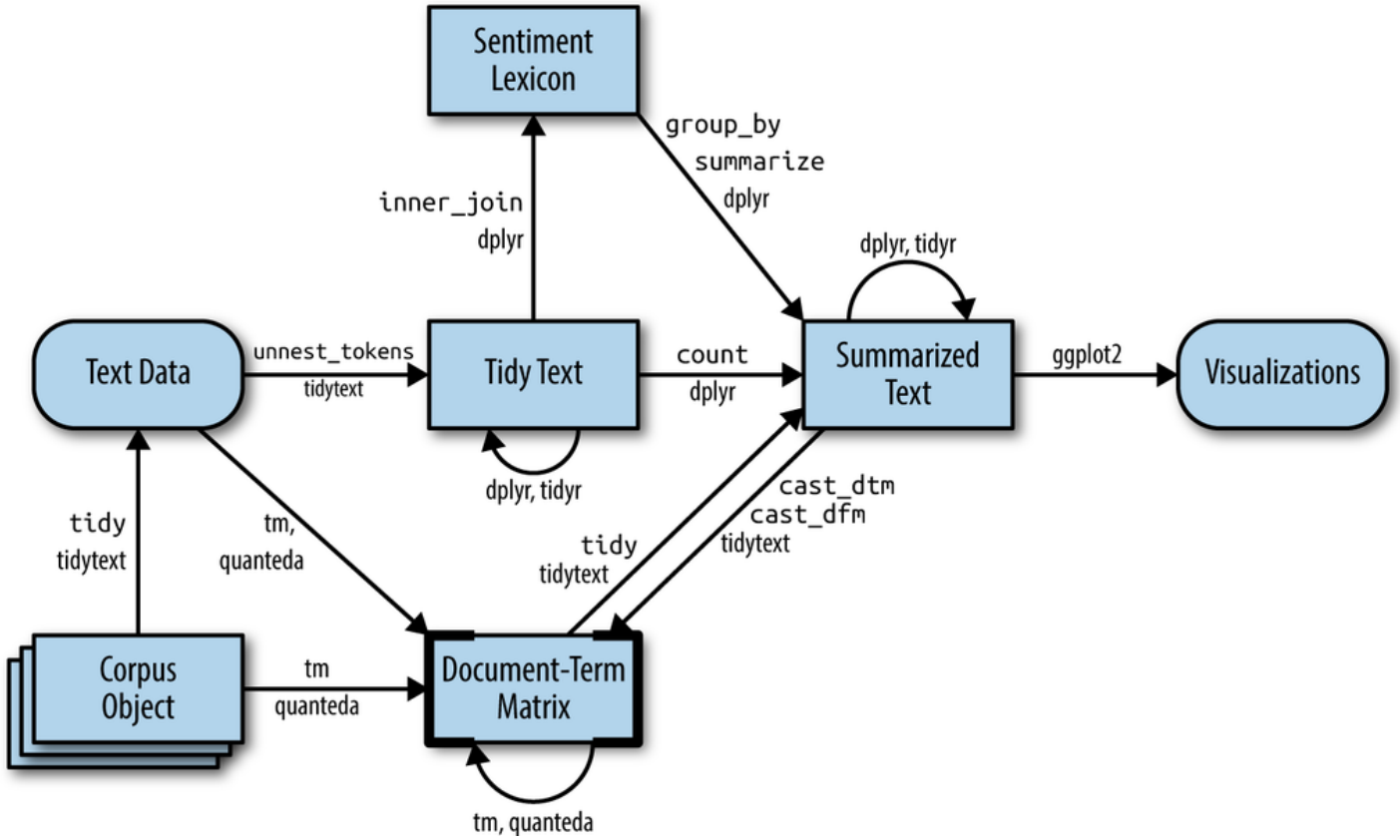
Geonwoo Ban

2021 4 7

앞선 챕터에서는 **tidy text structure**형식으로 배열된 텍스트를 분석했는데, 이러한 형식의 텍스트 데이터는 dplyr, tidyr 및 ggplot2와 같은 library를 사용하여 데이터를 탐색하고 시각화할 수 있다.

그러나 자연어 처리를 위한 기존의 R tools은 대부분 tidytext 패키지 외에는 tidy structure와 호환되지 않는다. 자연어 처리에는 input을 정돈되지 않은 데이터를 받아도 output에는 정돈된 결과로 내보내는 많은 패키지들이 있다. 이러한 패키지들은 텍스트 마이닝 애플리케이션에 유용해서, 많은 기존 텍스트 데이터셋은 이러한 형식(non-tidy format)에 따라 구성된다.

이번 장에서는 tidy text structure를 다른 중요한 패키지 및 데이터 구조와 연결하는 방법에 대해 논함으로써, 기존 텍스트 마이닝 패키지와 tidy tool을 결합하여 더 원활한 분석작업을 수행할 수 있게 하는것이 목적이다.



이 그림은 tidy 와 non-tidy data structure, 그리고 tidy tools와 untidy tools간에 분석 작업을 converting하는 것을 보여준다. 이번 장에서는 Document-Term Matrix를 정돈하는 과정과, 정돈된 행렬에 tidy data frame을 casting하는 과정에 초점을 맞출 것이다.

또한, raw text와 document metadata를 결합한 Corpus object들을 text data frame으로 정돈함으로써, 금융 관련 기사를 수집하여 분석하는 사례연구를 진행하는 방법으로 살펴볼 것이다.

5.1 Tidying a document-term matrix

텍스트 마이닝 패키지가 작동하는 가장 흔한 구조 중 하나는 **document-term matrix(DTM)**이다. 이러한 행렬은 다음과 같은 성질을 만족한다. + 각 행은 하나의 문서(ex. 도서나 논문)를 나타낸다. + 각 열은 하나의 용어를 나타낸다. + 각 값에는 (일반적으로) 해당 문서에서 해당 용어가 출현하는 횟수가 들어간다.

document와 term이 서로 하나의 쌍으로 이루어지는 경우는 거의 일어나지 않으므로 DTM은 일반적으로 sparse matrix가 된다. 이러한 객체를 행렬처럼 처리를 할 수도 있지만 더 효율적인 형식으로 저장할 수도 있다. 이번 장에서는 이러한 행렬에 대한 몇 가지 구현을 설명한다.

대부분의 텍스트 마이닝 패키지에 tidy data frame을 input으로 사용할 수 없듯이 DTM 객체를 tidy tool에서 바로 사용할 수는 없다. 따라서 tidytext 패키지는 두 가지 형식을 변환하는 두 개의 verbs를 제공한다. + `tidy()` 는 document-term matrix를 tidy data frame으로 바꾼다. + `cast()` 는 tidy data frame을 matrix로 변환해준다.

5.1.1 Tidying DocumentTermMatrix objects

가장 널리 사용되는 DTM 구현은 **tm**패키지의 DocumentTermMatrix class일 것이다. 사용 가능한 많은 텍스트 마이닝 데이터셋이 이 형식으로 제공된다. 예를 들어 topicmodels 패키지에 포함된 'Associated Press'라는 신문 기사 모음집을 고려해보자.

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.0.5
```

```
## Loading required package: NLP
```

```
data("AssociatedPress", package = "topicmodels")
AssociatedPress
```

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

이 데이터셋에는 문서(각 AP 기사)와 용어(별개의 단어들)가 포함되어 있다. 이 DTM은 99%가 sparse하다 (document-term 쌍의 99%가 0이라는 것을 의미). `Terms()` 함수를 사용해 문서의 용어에 access할 수 있다.

```
terms <- Terms(AssociatedPress)

head(terms)
```

```
## [1] "aaron"      "abandon"    "abandoned"  "abandoning" "abbott"
## [6] "abboud"
```

tidy tool로 이 데이터를 분석하려면 먼저 one-token-per-document-per-row 구조로 된 데이터 프레임으로 변환해야 한다. `tidy()` 함수는 non-tidy object를 가져와 tidy data frame으로 만든다.

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.4
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.4
```

```
ap_td <- tidy(AssociatedPress)

ap_td
```

```
## # A tibble: 302,031 x 3
##    document term       count
##       <int> <chr>      <dbl>
##  1        1 adding         1
##  2        1 adult          2
##  3        1 ago            1
##  4        1 alcohol        1
##  5        1 allegedly      1
##  6        1 allen          1
##  7        1 apparently     2
##  8        1 appeared       1
##  9        1 arrested       1
## 10        1 assault        1
## # ... with 302,021 more rows
```

이제 document, term, count 변수가 있는 tidy data structure로 변환되었다. 이러한 format은 dplyr, tidytext 및 ggplot2 패키지를 사용한 분석에 용이한 format이다. 예를 들어 2장에서 사용한 방법으로 이 신문 기사에 대한 sentiments analysis를 수행할 수 있다.

```
ap_sentiments <- ap_td %>%
  inner_join(get_sentiments("bing"), by = c(term = "word")) # 단어들에 대해 긍정과 부정의 이항분류로 나눈 bing lexicon을 사용.

ap_sentiments
```

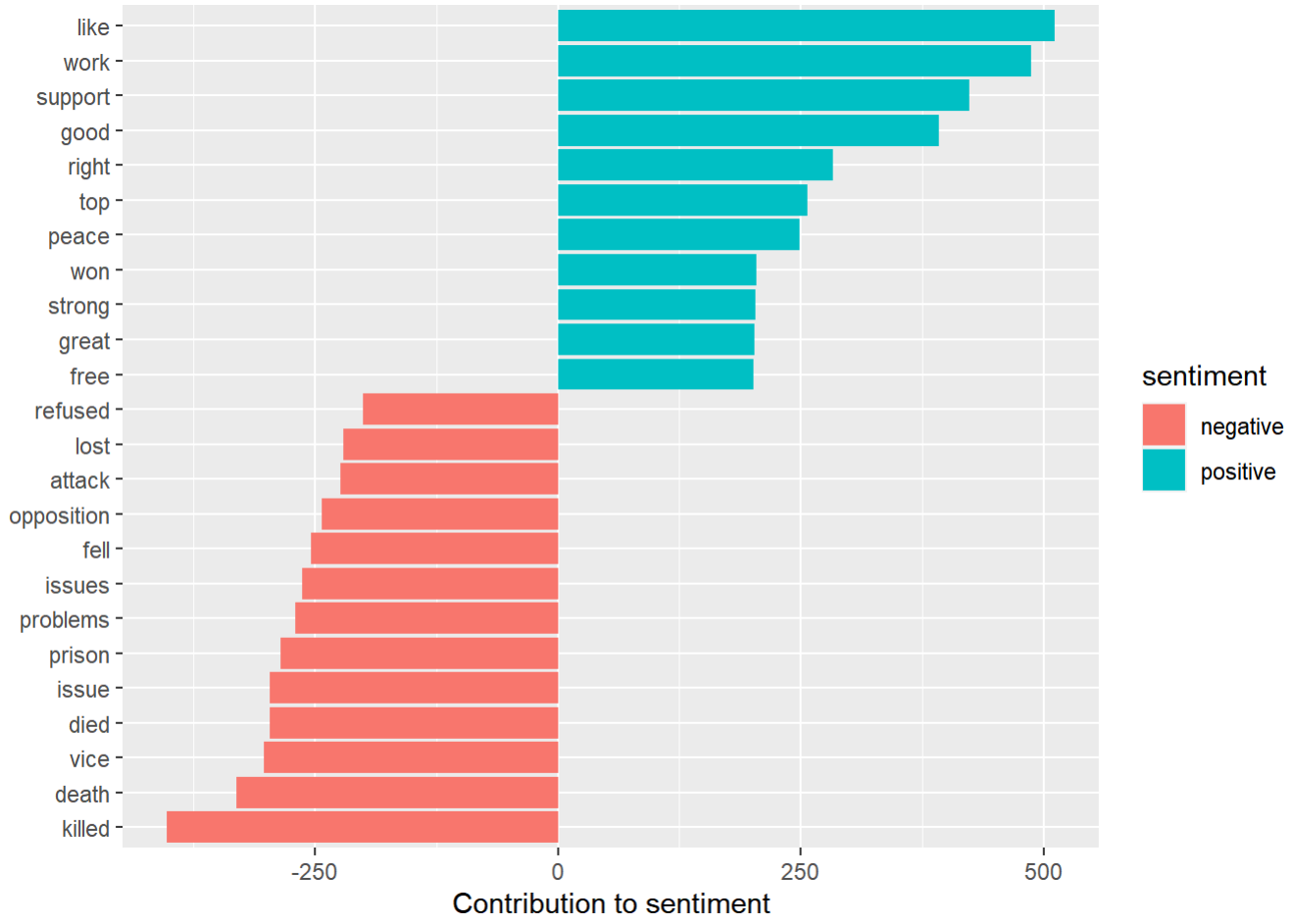
```
## # A tibble: 30,094 x 4
##    document term    count sentiment
##       <int> <chr>   <dbl> <chr>
##  1        1 assault     1 negative
##  2        1 complex     1 negative
##  3        1 death       1 negative
##  4        1 died        1 negative
##  5        1 good        2 positive
##  6        1 illness     1 negative
##  7        1 killed      2 negative
##  8        1 like        2 positive
##  9        1 liked       1 positive
## 10        1 miracle     1 positive
## # ... with 30,084 more rows
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
ap_sentiments %>%
  count(sentiment, term, wt = count) %>%
  ungroup() %>%
  filter(n >= 200) %>%                                  # 사용횟수가 200번 이상인 단어들에 대해,
  mutate(n = ifelse(sentiment == "negative", -n, n)) %>% # sentiment가 "negative"면 음수의 값으로 변환.
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(n, term, fill = sentiment)) +
  geom_col() +
  labs(x = "Contribution to sentiment", y = NULL)
```



sentiment analysis를 통해 Associate Press 기사에서 가장 자주 등장하는 긍정과 부정 단어들을 시각화 해본 결과, 가장 흔히 나온 긍정 단어에는 “like”, “work”, “support”, “good”이 포함되며, 부정 단어에는 “killed”, “death”, “vice”가 포함된다.

5.1.2 Tidying dfm objects

그 밖의 텍스트 마이닝 패키지들은 quanteda 패키지의 **document-feature matrix(dfm)** class와 같은 document-term matrix의 대체 format을 제공한다. 대통령 취임 연설문 데이터셋을 가지고 dfm으로 변환해보자.

```
data("data_corpus_inaugural", package = "quanteda")
inaug_dfm <- quanteda::dfm(data_corpus_inaugural, verbose = FALSE)
inaug_dfm
```

```
## Document-feature matrix of: 58 documents, 9,360 features (91.8% sparse) and 4 docvars.
##                  features
## docs              fellow-citizens  of the senate and house representatives :
##   1789-Washington               1  71 116      1  48     2               2 1
##   1793-Washington               0  11  13      0   2     0               0 1
##   1797-Adams                    3 140 163      1 130     0               2 0
##   1801-Jefferson                2 104 130      0  81     0               0 1
##   1805-Jefferson                0 101 143      0  93     0               0 0
##   1809-Madison                  1  69 104      0  43     0               0 0
##                  features
## docs              among vicissitudes
##   1789-Washington     1            1
##   1793-Washington     0            0
##   1797-Adams          4            0
##   1801-Jefferson      1            0
##   1805-Jefferson      7            0
##   1809-Madison        0            0
## [ reached max_ndoc ... 52 more documents, reached max_nfeat ... 9,350 more features ]
```

`tidy` 함수는 이러한 document-feature matrix에서도 tidy text structure으로 변환한다.

```
inaug_td <- tidy(inaug_dfm)
inaug_td
```

```
## # A tibble: 44,710 x 3
##    document        term            count
##    <chr>           <chr>           <dbl>
##  1 1789-Washington fellow-citizens     1
##  2 1797-Adams      fellow-citizens     3
##  3 1801-Jefferson  fellow-citizens     2
##  4 1809-Madison    fellow-citizens     1
##  5 1813-Madison    fellow-citizens     1
##  6 1817-Monroe     fellow-citizens     5
##  7 1821-Monroe     fellow-citizens     1
##  8 1841-Harrison   fellow-citizens    11
##  9 1845-Polk       fellow-citizens     1
## 10 1849-Taylor     fellow-citizens     1
## # ... with 44,700 more rows
```

각 취임식 연설에서 각 연설자별 공약이라던지 주요 단어들을 뽑아보는 것이 중요할 수 있기 때문에, 3장에서 하였던 것처럼 `bind_tf_idf()` 함수를 사용해 각 **term-speech**쌍의 tf-idf를 계산해 볼 수 있다.

```
inaug_tf_idf <- inaug_td %>%
  bind_tf_idf(term, document, count) %>%
  arrange(desc(tf_idf))

inaug_tf_idf
```

```
## # A tibble: 44,710 x 6
##    document        term        count      tf   idf tf_idf
##    <chr>           <chr>       <dbl>   <dbl> <dbl>  <dbl>
##  1 1793-Washington arrive          1 0.00680  4.06 0.0276
##  2 1793-Washington upbraidings     1 0.00680  4.06 0.0276
##  3 1793-Washington violated        1 0.00680  3.37 0.0229
##  4 1793-Washington willingly       1 0.00680  3.37 0.0229
##  5 1793-Washington incurring       1 0.00680  3.37 0.0229
##  6 1793-Washington previous        1 0.00680  2.96 0.0201
##  7 1793-Washington knowingly       1 0.00680  2.96 0.0201
##  8 1793-Washington injunctions     1 0.00680  2.96 0.0201
##  9 1793-Washington witnesses       1 0.00680  2.96 0.0201
## 10 1793-Washington besides         1 0.00680  2.67 0.0182
## # ... with 44,700 more rows
```

tf-idf를 계산한 이 데이터를 사용하여 링컨, 루즈벨트, 케네디, 오바마 대통령으로부터 주목할만한 취임 연설을 골라서 각 연설에 가장 주요단어로 보이는 것들을 시각화할 수 있다.

```
library(forcats)

dev.new(width=100, height=500, unit="in")

inaug_tf_idf %>%
  filter(document %in% c("1861-Lincoln", "1933-Roosevelt", "1961-Kennedy", "2009-Obama")) %>%
  group_by(document) %>% # 연설문별 그룹화.
  slice_max(tf_idf, n =9) %>% # 각 년도별 연설문에서 tf-idf값이 높은 9개의 단어만 선정
  ungroup() %>%
  ggplot(aes(tf_idf, fct_reorder(term, tf_idf), fill = document)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~document, ncol = 2, scales = "free") +
  labs(x = "tf-idf", y = NULL)
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.0.4
```

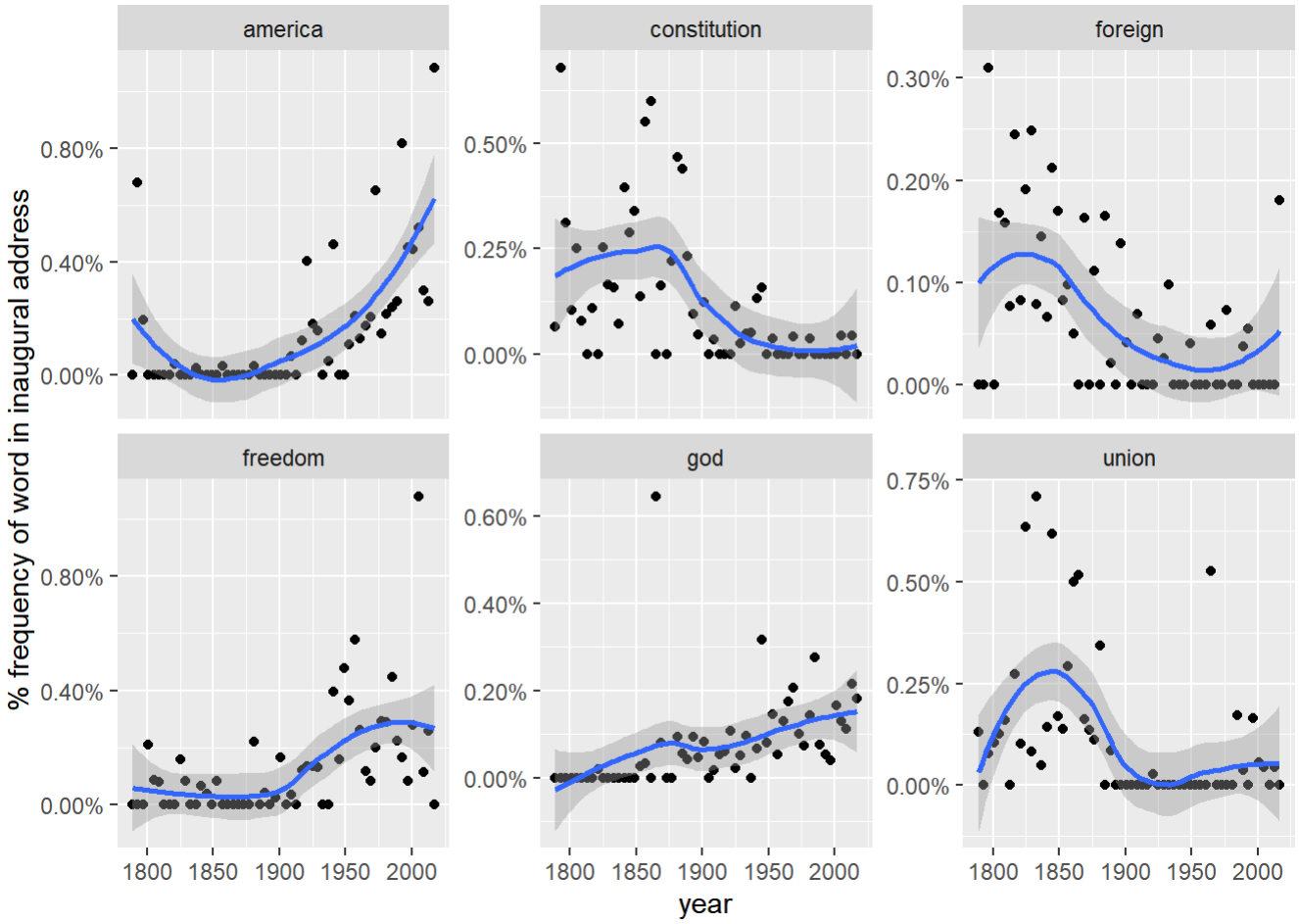
```
year_term_counts <- inaug_td %>%
  extract(document, "year", "(₩₩d+)", convert = TRUE) %>% # document변수에서 "-"를 기준으로 년도에 대한 정보만 가져왔음.
  complete(year, term, fill = list(count = 0)) %>% # NA인 관측치에 0을 넣어줌.
  group_by(year) %>%
  mutate(year_total = sum(count)) # 년도별 총 단어 사용 수 계산.

year_term_counts
```

```
## # A tibble: 542,880 x 4
## # Groups:   year [58]
##     year term  count year_total
##    <int> <chr> <dbl>      <dbl>
##  1  1789 "'"       0       1537
##  2  1789 "-"       1       1537
##  3  1789 "!"       0       1537
##  4  1789 "₩""      2       1537
##  5  1789 "$"       0       1537
##  6  1789 "("       1       1537
##  7  1789 ")"       1       1537
##  8  1789 ","      70       1537
##  9  1789 "."      23       1537
## 10  1789 ":"       1       1537
## # ... with 542,870 more rows
```

다른 년도의 연설문에는 있지만 특정 년도의 연설문에는 생략된 단어들을 모두 표시하기 위해 complete함수를 사용하여 NA를 대체하였다. 이후 특정 단어들에 대해 시간이 지남에 따라 빈도가 어떻게 변하는지를 확인할 수 있다.

```
year_term_counts %>%
  filter(term %in% c("god", "america", "foreign", "union", "constitution", "freedom")) %>%
  ggplot(aes(year, count / year_total)) +
  geom_point() +
  geom_smooth() +
  facet_wrap(~ term, scales = "free_y") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(y = "% frequency of word in inaugural address")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



- 연설문에 사용된 특정 단어의 비율이 시간에 지남에 따라 변하는 과정을 그린 그림이다.
- 이 그림을 통해 먼저 미국 대통령들이 미국을 “Union”으로 언급하는 것 보단 “America”로 언급을 하는 것을 확인할 수 있다.
- 또한 “Constitution(헌법)”과 “foreign(외국)”국가들에 대해서 언급이 줄게되며, “freedom”과 “god”을 더 많이 언급하는 것으로 보인다.

이렇게 문서가 tidy structure가 아니더라도 변환을 통해 여러 tidy tool들을 사용하여 분석하는 방법을 보여준다.

-Untidy text structure +Document-Term Matrix(DTM) : 문서별 포함된 단어에 대해 sparse하게 만들어진 matrix
+Document-Feature Matrix(dfm) : 문서별 사용된 모든 문자들에 대해 sparse하게 만들어진 matrix

5.2 Casting tidy text data into a matrix

앞서 untidy data structure에서 tidy structure로 바꾸는 방법을 확인하였다. 이번에는 반대로 tidy structure에서 untidy structure로 변환하기 위한 `cast_` 로 시작되는 함수들을 확인해보자.

예를 들면 tidy AP dataset을 가져와서 `cast_dtm()` 을 사용해 document-term matrix로 다시 casting할 수 있다.

```
ap_td
```

```
## # A tibble: 302,031 x 3
##    document term       count
##       <int> <chr>      <dbl>
##  1        1 adding         1
##  2        1 adult          2
##  3        1 ago            1
##  4        1 alcohol        1
##  5        1 allegedly      1
##  6        1 allen          1
##  7        1 apparently     2
##  8        1 appeared       1
##  9        1 arrested       1
## 10        1 assault        1
## # ... with 302,021 more rows
```

```
ap_td %>%
  cast_dtm(document, term, count)
```

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

마찬가지로 document-feature matrix로도 변환이 가능하다.

```
ap_td %>%
  cast_dfm(document, term, count)
```

```
## Document-feature matrix of: 2,246 documents, 10,473 features (98.7% sparse).
##     features
## docs adding adult ago alcohol allegedly allen apparently appeared arrested
##    1      1     2   1       1         1     1          2        1        1
##    2      0     0   0       0         0     0          0        1        0
##    3      0     0   1       0         0     0          0        1        0
##    4      0     0   3       0         0     0          0        0        0
##    5      0     0   0       0         0     0          0        0        0
##    6      0     0   2       0         0     0          0        0        0
##     features
## docs assault
##    1       1
##    2       0
##    3       0
##    4       0
##    5       0
##    6       0
## [ reached max_ndoc ... 2,240 more documents, reached max_nfeat ... 10,463 more features ]
```

5.3 Tidying corpus objects with metadata

토큰화를 하기 전에 문서 모음집들을 저장해 두도록 설계된 데이터 구조들이 있는데, 이것들을 **Corpus(말뭉치)**라고 부른다. 여기에는 ID, 날짜/시간, 제목 또는 각 문서의 언어가 포함될 수 있는 메타데이터 옆에 텍스트가 저장되는 구조이다.

예를 들어 tm 패키지에는 Reuters 통신사의 50개 기사가 들어 있는 acq라는 Corpus가 제공된다.

```
data("acq")

acq
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 50
```

```
acq[[1]] # 첫번째 문서
```

```
## <<PlainTextDocument>>
## Metadata:  15
## Content:  chars: 1287
```

Corpus object는 list와 같은 구조로 되너있으며 각 항목에는 텍스트와 메타데이터가 모두 포함된다. 이러한 Corpus object를 사용하는 것은 문서를 편리하게 저장하는 방법이기는 하지만 tidy tool들로 처리하기에는 적합하지 않은 구조이다. 따라서 tidy tool을 사용하기 위해 tidy structure로 바꿔보자.

```
acq_td <- tidy(acq)

acq_td
```

```
## # A tibble: 50 x 16
##    author  datetimestamp       description heading  id    language origin topics
##    <chr>   <dttm>              <chr>       <chr>    <chr> <chr>    <chr>  <chr>
##  1 <NA>    1987-02-27 00:18:06 ""          COMPUTE~ 10    en       Reute~ YES
##  2 <NA>    1987-02-27 00:19:15 ""          OHIO MA~ 12    en       Reute~ YES
##  3 <NA>    1987-02-27 00:49:56 ""          MCLEAN'~ 44    en       Reute~ YES
##  4 By Cal~ 1987-02-27 00:51:17 ""          CHEMLAW~ 45    en       Reute~ YES
##  5 <NA>    1987-02-27 01:08:33 ""          <COFAB ~ 68    en       Reute~ YES
##  6 <NA>    1987-02-27 01:32:37 ""          INVESTM~ 96    en       Reute~ YES
##  7 By Pat~ 1987-02-27 01:43:13 ""          AMERICA~ 110   en       Reute~ YES
##  8 <NA>    1987-02-27 01:59:25 ""          HONG KO~ 125   en       Reute~ YES
##  9 <NA>    1987-02-27 02:01:28 ""          LIEBERT~ 128   en       Reute~ YES
## 10 <NA>    1987-02-27 02:08:27 ""          GULF AP~ 134   en       Reute~ YES
## # ... with 40 more rows, and 8 more variables: lewissplit <chr>,
## #   cgisplit <chr>, oldid <chr>, places <named list>, people <lgl>, orgs <lgl>,
## #   exchanges <lgl>, text <chr>
```

이렇게 tidy structure로 바꾼다음 unnest_tokens()와 함께 사용하여 가장 빈도수가 높은 단어들 및 tf-idf값이 높은 단어들을 확인할 수 있다.

```
acq_tokens <- acq_td %>%
  select(-places) %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by = "word")
```

```
## Warning: Outer names are only allowed for unnamed scalar atomic inputs
```

```
acq_tokens %>%
  count(word, sort=T)
```

```
## # A tibble: 1,566 x 2
##    word         n
##    <chr>    <int>
##  1 dlrs       100
##  2 pct         70
##  3 mln         65
##  4 company     63
##  5 shares      52
##  6 reuter      50
##  7 stock       46
##  8 offer       34
##  9 share       34
## 10 american    28
## # ... with 1,556 more rows
```

```
acq_tokens %>%
  count(id, word) %>%
  bind_tf_idf(word, id, n) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 2,853 x 6
##    id    word         n     tf   idf tf_idf
##    <chr> <chr>    <int>  <dbl> <dbl>  <dbl>
##  1 186   groupe       2 0.133   3.91  0.522
##  2 128   liebert      3 0.130   3.91  0.510
##  3 474   esselte      5 0.109   3.91  0.425
##  4 371   burdett      6 0.103   3.91  0.405
##  5 442   hazleton     4 0.103   3.91  0.401
##  6 199   circuit      5 0.102   3.91  0.399
##  7 162   suffield     2 0.1     3.91  0.391
##  8 498   west         3 0.1     3.91  0.391
##  9 441   rmj          8 0.121   3.22  0.390
## 10 467   nursery      3 0.0968  3.91  0.379
## # ... with 2,843 more rows
```