

Topic Modeling

Introduction:

This report explores topic modeling on a dataset of movie plots. By utilizing Latent Dirichlet Allocation (LDA), we aim to identify thematic clusters within the text data, revealing underlying topics based on word distributions. The final output includes interpretations of each topic and visualizations to illustrate the findings.

Library Imports:

To perform the topic modeling and visualize the results, several R packages are necessary. These packages collectively enable text preprocessing, LDA model creation, topic selection, and result visualization. Below are the libraries used

```
library(topicmodels)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v dplyr      1.1.4      v readr      2.1.5
v forcats    1.0.0      v stringr    1.5.1
v ggplot2    3.5.1      v tibble     3.2.1
v lubridate  1.9.3      v tidyr      1.3.1
v purrr      1.0.2
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()     masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```
library(tm)
```

Loading required package: NLP

Attaching package: 'NLP'

The following object is masked from 'package:ggplot2':

annotate

```
library(SnowballC)
library(ldatuning)
library(wordcloud)
```

Loading required package: RColorBrewer

```
library(ggplot2)
library(tidytext)
```

Importing the Dataset:

The dataset used in this analysis is loaded from a CSV file containing movie plot descriptions. Here, we load the data and inspect the column names to ensure that the dataset structure is as expected.

```
movie_data <- read.csv("movie_plots_with_genres.csv")
```

```
colnames(movie_data)
```

```
[1] "row"          "Movie.Name" "Genre"      "Plot"
```

Text Preprocessing-

To prepare the movie plot descriptions for analysis, we convert the text data into a corpus. This corpus will undergo a series of transformations, including tokenization, lowercasing, punctuation removal, and stopwords filtering, to ensure the text is suitable for topic modeling.

1. **Convert to Lowercase:** All text was converted to lowercase to eliminate case sensitivity, so that words like “Hero” and “hero” are treated as the same token.

2. **Remove Punctuation:** Punctuation marks were removed, as they do not contribute to the meaning of words and can add unnecessary complexity to the model.
3. **Remove Numbers:** Numerical values were stripped from the text, as they often do not carry semantic meaning relevant to the topics we aim to identify.
4. **Remove Stopwords:** Commonly used words (e.g., “and,” “the,” “is”) that do not carry significant meaning were removed using a list of English stopwords. This step helps to reduce noise in the data and ensures that the model focuses on content-specific words rather than function words.
5. **Stemming:** Stemming reduces words to their root forms, so variations like “running” and “runner” are reduced to “run.” This step helps to group words with similar meanings, reducing the dimensionality of the text data and improving model focus on core terms rather than their inflections.

```
corpus <- VCorpus(VectorSource(movie_data$Plot))
```

```
# 1. Convert to lowercase
corpus <- tm_map(corpus, content_transformer(tolower))
empty_after_tolower <- sapply(corpus, function(doc) { length(doc$content) == 0 })
cat("Empty documents after tolower:", sum(empty_after_tolower), "\n")
```

Empty documents after tolower: 0

The `echo: false` option disables the printing of code (only output is displayed).

```
# 2. Remove punctuation
corpus <- tm_map(corpus, removePunctuation)
empty_after_punctuation <- sapply(corpus, function(doc) { length(doc$content) == 0 })
cat("Empty documents after punctuation removal:", sum(empty_after_punctuation), "\n")
```

Empty documents after punctuation removal: 0

```
# 3. Remove numbers
corpus <- tm_map(corpus, removeNumbers)
empty_after_numbers <- sapply(corpus, function(doc) { length(doc$content) == 0 })
cat("Empty documents after number removal:", sum(empty_after_numbers), "\n")
```

Empty documents after number removal: 0

```
# 4. Remove stopwords
corpus <- tm_map(corpus, removeWords, stopwords("english"))
empty_after_stopwords <- sapply(corpus, function(doc) { length(doc$content) == 0 })
cat("Empty documents after stopwords removal:", sum(empty_after_stopwords), "\n")
```

Empty documents after stopwords removal: 0

```
# 5. Stemming
corpus <- tm_map(corpus, stemDocument)
empty_after_stemming <- sapply(corpus, function(doc) { length(doc$content) == 0 })
cat("Empty documents after stemming:", sum(empty_after_stemming), "\n")
```

Empty documents after stemming: 0

```
# Final check for empty documents
empty_docs_final <- sapply(corpus, function(doc) { length(doc$content) == 0 })
cat("Total empty documents in final corpus:", sum(empty_docs_final), "\n")
```

Total empty documents in final corpus: 0

After each preprocessing step, we check for empty documents to ensure that no movie plot descriptions are entirely removed during transformations. This step is essential to retain meaningful data in the corpus for accurate topic modeling

```
# Remove any empty documents if needed
corpus <- corpus[!empty_docs_final]
```

Document Term Matrix Creation:

After preprocessing the text, we convert the corpus into a Document-Term Matrix (DTM). The DTM represents the frequency of each term across the documents, providing a structured format that the topic modeling algorithm can process. Each row in the DTM corresponds to a document, and each column represents a unique term, with values indicating term frequency in each document.

```
# Create Document-Term Matrix (DTM) with term frequency weighting
dtm <- DocumentTermMatrix(corpus)
```

Choosing the Optimal Number of Topics:

Choosing an appropriate number of topics is critical for generating meaningful and interpretable topics. We employ the `ldatuning` package to evaluate several metrics—CaoJuan2009, Arun2010, and Deveaud2014—across a range of topic numbers. These metrics provide different perspectives on model coherence, helping us select a topic number that balances interpretability and data fit.

The `FindTopicsNumber` function runs multiple LDA models with different topic counts (from 2 to 20 in increments of 2) and evaluates each model's performance based on the selected metrics. This process allows us to identify a topic count where the metrics stabilize or reach optimal values, suggesting a suitable number of topics.

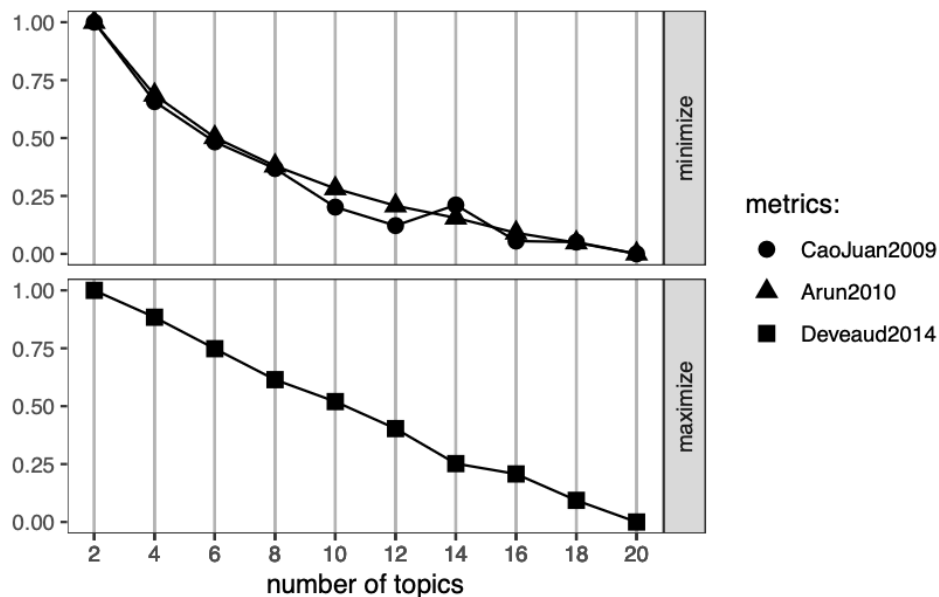
```
result <- FindTopicsNumber(
  dtm,
  topics = seq(2, 20, by = 2),
  metrics = c("CaoJuan2009", "Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 123),
  verbose = TRUE
)
```

```
fit models... done.
calculate metrics:
  CaoJuan2009... done.
  Arun2010... done.
  Deveaud2014... done.
```

```
FindTopicsNumber_plot(result)
```

Warning: The ``<scale>`` argument of ``guides()`` cannot be ``FALSE``. Use "none" instead as of ggplot2 3.3.4.

i The deprecated feature was likely used in the `ldatuning` package.
Please report the issue at <https://github.com/nikita-moor/ldatuning/issues>.



The plot above visualizes the evaluation of different metrics—CaoJuan2009, Arun2010, and Deveaud2014 across a range of topic numbers. Each metric offers a different criterion for determining the optimal number of topics:

- **CaoJuan2009** is minimized when topics are well separated, and coherence is high. We look for a point where the metric stabilizes at a lower value.
- **Arun2010** follows a similar pattern, where lower values indicate better topic separation, allowing us to select a stable point.
- **Deveaud2014** is maximized, so we look for higher values indicating cohesive topics.

By examining the plots, we identify a range where the metrics start to level off, which indicates a suitable balance between model complexity and interpretability. Based on the trends observed in this plot, we selected 10 topics as the optimal number for our model, balancing stability across metrics.

```
# Set the optimal number of topics
optimal_k <- 10

# Run the LDA model with the chosen number of topics
lda_model <- LDA(dtm, k = optimal_k, control = list(seed = 1234))
```

Running the LDA Model to interpret the Model

With the optimal number of topics ($k=10$) selected, we ran the LDA model on the Document-Term Matrix to uncover the underlying themes in the dataset.

```
# Display the top 10 terms for each topic
top_terms <- terms(lda_model, 10)
print(top_terms)
```

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
[1,]	"take"	"gang"	"fight"	"love"	"will"	"life"	"world"
[2,]	"will"	"ranch"	"life"	"young"	"find"	"world"	"game"
[3,]	"one"	"kill"	"red"	"life"	"one"	"alien"	"team"
[4,]	"drug"	"town"	"will"	"forc"	"new"	"will"	"wrestl"
[5,]	"two"	"get"	"two"	"girl"	"time"	"get"	"time"
[6,]	"race"	"outlaw"	"get"	"one"	"world"	"find"	"new"
[7,]	"christma"	"sheriff"	"jack"	"find"	"get"	"day"	"will"
[8,]	"fight"	"rancher"	"back"	"fall"	"life"	"love"	"jake"
[9,]	"can"	"murder"	"one"	"new"	"take"	"young"	"match"
[10,]	"world"	"jim"	"brother"	"daughter"	"crew"	"take"	"stori"

	Topic 8	Topic 9	Topic 10
[1,]	"will"	"film"	"bill"
[2,]	"life"	"danc"	"hors"
[3,]	"take"	"sport"	"two"
[4,]	"one"	"world"	"get"
[5,]	"man"	"featur"	"town"
[6,]	"stori"	"game"	"one"
[7,]	"get"	"one"	"find"
[8,]	"find"	"seri"	"time"
[9,]	"world"	"new"	"film"
[10,]	"war"	"will"	"father"

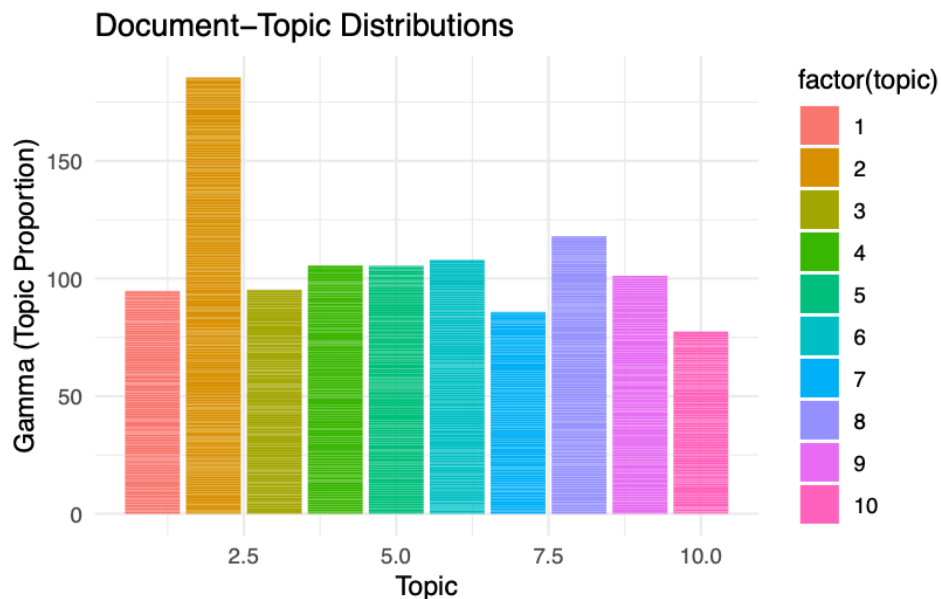
- Based on the top terms for each topic, we can make preliminary interpretations of what each topic represents-
- **Topic 1:** Terms like “take,” “bill,” and “father” suggest themes around **family dynamics or personal quests**, potentially involving relationships or responsibilities.
- **Topic 2:** Words such as “gang,” “ranch,” “town,” and “sheriff” indicate a **Western or crime-related theme**, likely focusing on rural settings and law enforcement.
- **Topic 3:** With terms like “life,” “young,” and “love,” this topic could represent **romantic or coming-of-age themes**, focusing on growth, relationships, and self-discovery.

- **Topic 4:** Words like “kill,” “war,” and “alien” hint at a **sci-fi or action genre**, potentially involving battles, otherworldly elements, or high-stakes conflicts.
- **Topic 5:** Terms like “life,” “world,” and “crew” suggest a focus on **exploration or adventure**, possibly involving teams or journeys to unknown places.
- **Topic 6:** Words such as “find,” “game,” “time,” and “day” may relate to **sports or competition**, where time and strategy play crucial roles.
- **Topic 7:** With terms like “world,” “new,” “team,” and “one,” this topic could focus on **team-based adventures or exploration**.
- **Topic 8:** Words like “film,” “dance,” and “feature” suggest a possible focus on **art, performance, or creative expression**.
- **Topic 9:** Terms such as “new,” “match,” and “life” might be linked to **self-discovery or transformative journeys**, with individuals undergoing significant changes.
- **Topic 10:** Includes words like “father,” “take,” and “family,” indicating themes around **family relationships** and possibly **generational stories or heritage**.

```
# Convert the LDA model to a tidy format
gamma_df <- tidy(lda_model, matrix = "gamma")
```

A Few Visualisations:

```
# Plot document-topic distributions
ggplot(gamma_df, aes(topic, gamma, fill = factor(topic))) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Document-Topic Distributions", x = "Topic", y = "Gamma (Topic Proportion)")
```

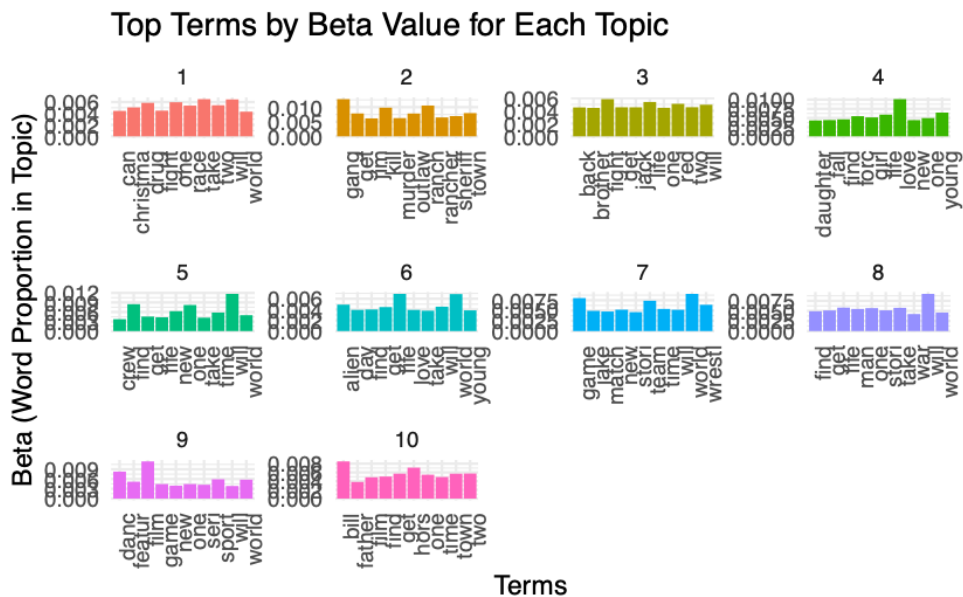
The bar chart above visualizes the distribution of topics across the entire set of movie plots. Each bar represents one of the ten topics, and the height of each bar reflects the proportion of documents (movie plots) that align with that topic.

- **Interpretation of Gamma Values:** In the context of topic modeling, the gamma value represents the probability that a document is associated with a given topic. Higher gamma values for a topic indicate a stronger presence or prevalence of that theme within the dataset.
- **Topic Prevalence:** This plot helps identify which topics are more dominant within the dataset. For instance, Topic 2 (indicated by the orange bar) has a high prevalence, suggesting it is a common theme among the movie plots. On the other hand, topics with shorter bars appear less frequently.

The below plot shows the distribution of words within each topic, helping illustrate which words are most associated with each topic. This plot shows the top terms for each topic based on beta values, where beta represents the likelihood of each word given a topic. It's useful for understanding the distinct terms that define each topic.

```
beta_df <- tidy(lda_model, matrix = "beta")
top_beta_terms <- beta_df %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
```

```
ggplot(top_beta_terms, aes(term, beta, fill = factor(topic))) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Top Terms by Beta Value for Each Topic", x = "Terms", y = "Beta (Word Proportion in Topic)")
```



The below plot is a heatmap. This heatmap shows the topic proportions for each document, helping visualize which topics dominate in different documents. This heatmap allows you to see which topics are most prominent across documents, giving a quick overview of topic distribution across the dataset.

```
library(reshape2)
```

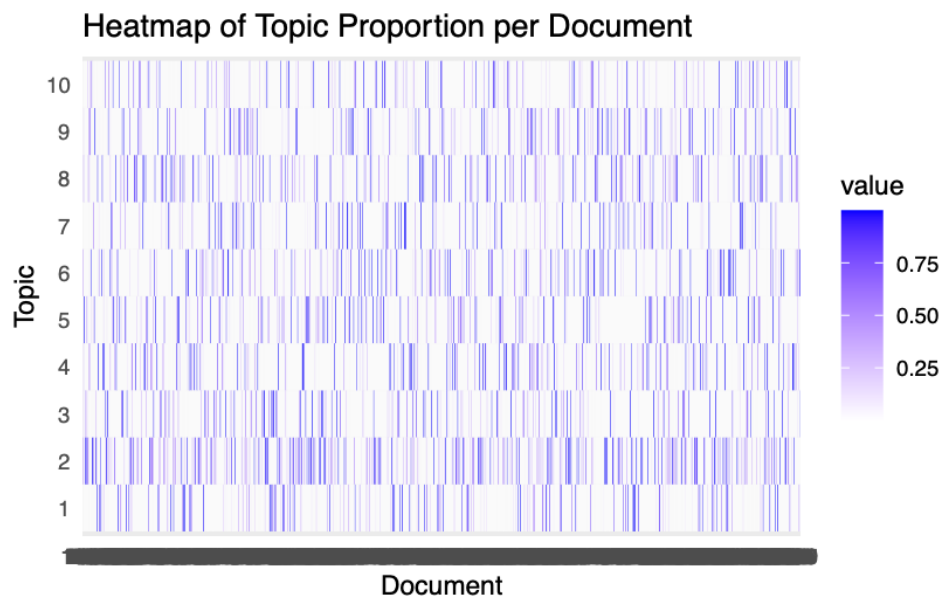
Attaching package: 'reshape2'

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

smiths

```
gamma_df <- tidy(lda_model, matrix = "gamma")
gamma_wide <- dcast(gamma_df, document ~ topic, value.var = "gamma")

ggplot(melt(gamma_wide, id = "document"), aes(document, variable, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Heatmap of Topic Proportion per Document", x = "Document", y = "Topic") +
  theme_minimal()
```



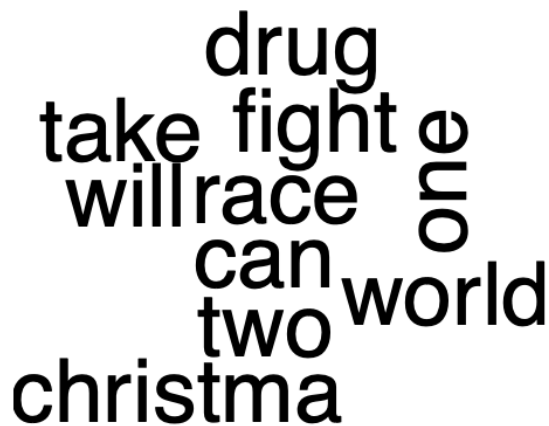
Word Clouds:

The word clouds above visually represent the most significant terms for each of the ten topics derived from our topic modeling analysis. In each word cloud, the size of each word corresponds to its importance within the topic, with larger words representing higher relevance or frequency in the documents associated with that topic.

```
# Generate word clouds for each topic
for (i in 1:optimal_k) {
  wordcloud(words = top_terms[, i], scale = c(3, 0.5), max.words = 50)
}
```

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation  
drops documents
```


```
Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,  
tm::stopwords())): transformation drops documents
```



A word cloud visualization showing the following words: drug, take, fight, will, race, can, two, world, and christma. The words are arranged in a roughly triangular shape, with 'drug' at the top, 'take fight' in the middle, and 'christma' at the bottom. The words are in a black, sans-serif font.

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation  
drops documents
```

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation  
drops documents
```



A word cloud visualization showing the following words: get jim, sheriff, kill gang, and outlaw. The words are arranged in a roughly triangular shape, with 'get jim' at the top, 'sheriff' in the middle, and 'kill gang' at the bottom. The words are in a black, sans-serif font.

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation  
drops documents
```

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation  
drops documents
```

brother
get will
red two
fight life

Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents

daughter
love find
new life
one fall
for girl

Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents

one time
will find
new life
crew det

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
drops documents
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
drops documents
```

world
get love
will alien
young find

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
drops documents
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
drops documents
```

jake
stori time
will world
wrestl
demonow

```
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
drops documents
Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
drops documents
```

world : take
 story find war
 s get
 man life
 ...!!!

Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents
 Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents

new seri
 danc
 film world
 game
 natur
 will

Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents
 Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation drops documents

time town
 find hors
 get bill
 father one
 film two