



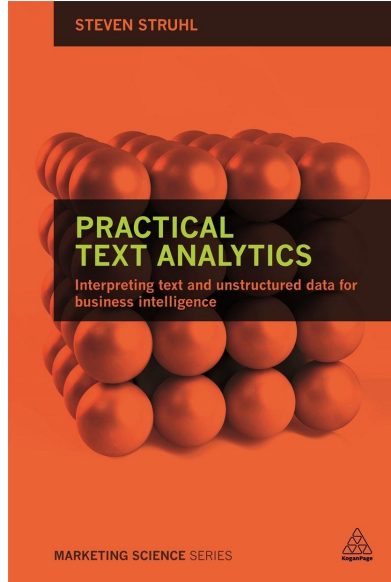
# TOKENIZING TEXT

R, stringr & tidr

# CONTENT

---

- Filter or Sample Data
- Clean and Normalize Text
- Split Text into Tokens
- Remove Stop Words
- Enrich Tokens (Stemming, Lemmatization, Part-of-Speech Tagging)

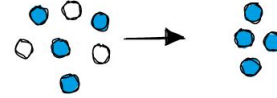


Struhl, Steven M. (2015): *Practical Text Analytics: Interpreting text and unstructured data for business intelligence*. London, UK, Philadelphia, PA: Kogan Page (Marketing science series).

# Tokenization

Five steps to impose a structure on text

1. Filter or sample data



2. Clean and normalize text

"@all: This is the best course ever!!"

becomes

"this is the best course ever"

3. Split text into tokens

["this", "is", "the", "best", "course", "ever"]

4. Remove stop words

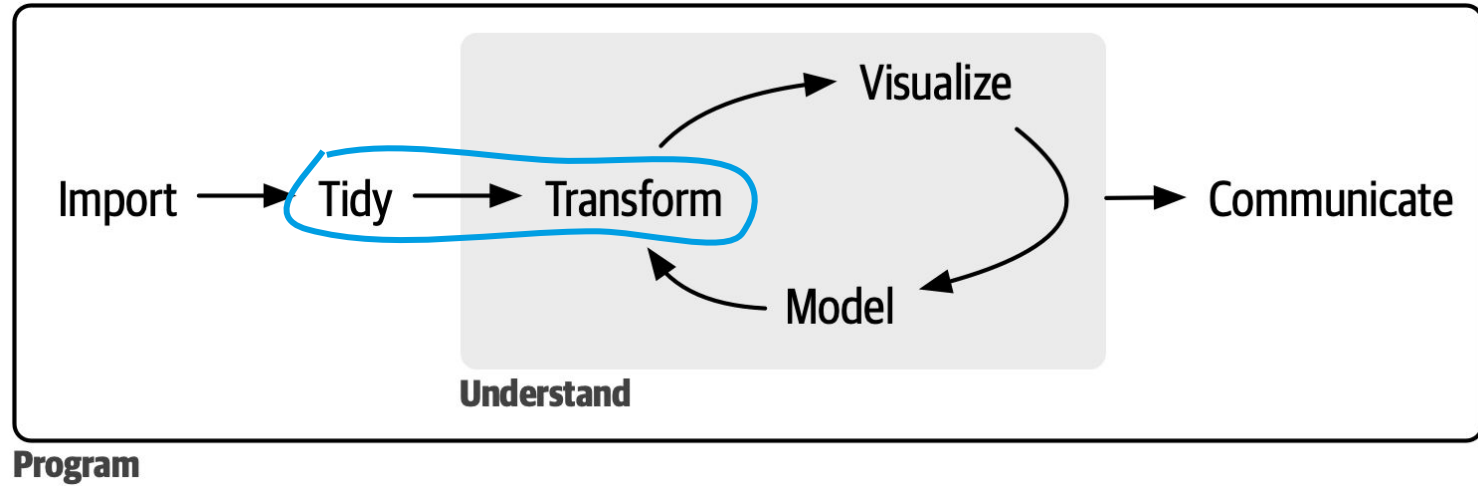
["is", "best", "course", "ever"]

5. Enrich tokens (lemmatization, stemming, part-of-speech)

```
["be" : [verb],  
 "best": [adj],  
 "course": [noun, obj],  
 "ever": [temporal] ]
```

## WHERE ARE WE?

---



Source: Wickham, Hadley, and Garrett Grolemund. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. First edition, O'Reilly, 2016. URL: <https://r4ds.hadley.nz/diagrams/data-science/base.png>

# **FILTER OR SAMPLE DATA**

`filter, slice_sample`

# DATA IS TOO LARGE?

## WHAT OPTIONS DO WE HAVE?

---

With analyzing data on our laptop, we are limited in RAM and CPU.  
What if the data is just too big?

- Reduce the data by filtering out irrelevant records (**filter**)
- Reduce the data through sampling and proceeding with a smaller portion (**slice\_sample**)
- Pay for more compute resources, e.g., using a Spark cluster on Databricks
- Logon to the high performance compute (HPC) cluster from the university



Created with Bing Image Creator 2023

## FILTER OR SAMPLE DATA

### FILTER

---

Reduce the data set before further analysis with **filter**:

```
tweets_filtered <-  
  tweets |>  
  filter(year(created_at) == 2023) |>  
  filter(lang == "de") |>  
  select(id, created_at, screen_name, text, is_retweet)
```



When filtering is not possible, we can still use sampling with `slice_sample` from `{dplyr}`:

```
tweets |>  
  slice_sample(n = 1000)
```

```
tweets |>  
  slice_sample(prop = 0.1)
```

**CLEAN AND  
NORMALIZE TEXT**

Text often contains irrelevant characters or character sequences that are of no value for analysis.

**Cleaning** text means to remove them from text. This can include:

- Punctuation
- Special characters, e.g., @%&#/
- Invisible characters, e.g., \n, \r, \t, or multiple spaces
- Specific character sequences, e.g., URLs, user mentions, hashtags

**Normalization** involves converting all characters to lowercase for better comparison.

## CLEAN AND NORMALIZE

### EXAMPLE

---

"@all This is the best course ever! #bigdata #nlp 🙌"



"this is the best course ever"

## CLEAN (1/2)

### EXAMPLE TWEETS

---

Remove character sequences from tweets with `str_remove_all`, `str_replace_all`, and `str_trim`:

```
tweets_clean <-  
  tweets_filtered |>  
  mutate(text = str_remove_all(text, "@\\w+")) |>  
  mutate(text = str_remove_all(text, "#\\w+")) |>  
  mutate(text = str_remove_all(text, "https?:\\/\\/\\S+")) |>  
  mutate(text = str_remove_all(text, "[[:punct:]]")) |>  
  mutate(text = str_replace_all(text, "\\s{2,}", " ")) |>  
  mutate(text = str_trim(text))
```

## CLEAN (2/2)

### EXAMPLE TWEETS

---

With regular expressions, we can remove all kinds of characters:

```
emojis <- "[\U0001F600-\U0001F64F\U0001F300-\U0001F5FF+]"
```

```
tweets |>
```

```
  mutate(text = str_remove_all(text, emojis))
```

## NORMALIZE

---

Convert the text to lowercase with `str_to_lower`:

```
tweets |>  
  mutate(text = str_to_lower(text))
```

**SPLIT TEXT INTO TOKENS**



Tokenization is one way to impose structure on otherwise unstructured text data.

- The assumption is that text is made of words (or tokens) separated by a space
- By splitting text into words (or tokens), we create a column with:
  - Atomic values → one word per column
  - A discrete range of values → the vocabulary used in the text data
- Methods like **filter**, **group\_by**, **count** and the like can be applied → **Analysis is possible**
- Beware of the limits!

"this is the best course ever"



"this"  
"is"  
"the"  
"best"  
"course"  
"ever"

## SPLIT TEXT

---

We can split text based on a separator and expand the result into rows with `separate_longer_delim`:

```
tweets_tokenized <-  
  tweets_clean |>  
  tidyr::separate_longer_delim(text, " ") |>  
  rename(word = text)
```

**REMOVE STOP WORDS**

## REMOVE STOP WORDS

---

Many words appear frequently in text but have very little meaning for analysis.

- Filter out words with little contribution to content, sentiment, meaning etc.
- Only then can we uncover the interesting words and their usage
- We can filter stop words based on a simple list and the **anti\_join** function
  - [List with English stop words](#) (~670)
  - [List with German stop words](#) (~620)

## REMOVE STOP WORDS

---

The `anti_join` is the opposite of a join and removes any rows with a match:

```
stop <- read_csv("data/stopwords_german.csv")
```

```
tweets_tokenized |>  
  anti_join(stop, by = "word") |>  
  count(word, sort = TRUE)
```

text

@all This is the best course ever! #bigdata #nlp 🙌

Cleaning and normalization

text

this is the best course ever

Tokenization

word

this

is

the

best

course

ever

Stop  
words

word

best

course

ever

# ENRICH TOKENS

Stemming, Lemmatization, Part-of-Speech



Now that we have a column with word, we can add more metadata, such as:

- What is the word's stem? (eats → eat, sitting → sit) ([stemming](#))
- What is the base form of the word (is → be, mice → mouse, best → good) ([lemmatization](#))
- What type of word is it (noun, verb, adjective...) ([part-of-speech tagging](#))
- What role does the word play in its context? ([contextual dependencies](#))

We could do the first three with the same rule-based approach as for the stop words (the last won't work that way). We'll see that [probabilistic models from machine learning](#) are much better at this.