



# **RULE-BASED TEXT CLASSIFICATION**

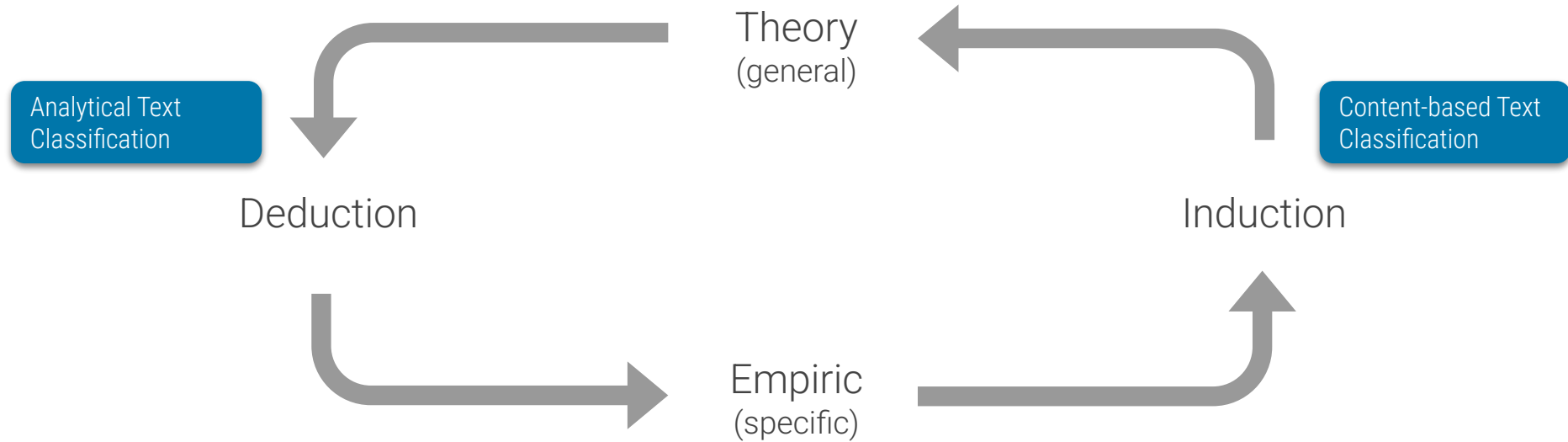
with R

- Inductive and Deductive
- Preparation: Tokenize Text
- Deductive Topic Classification
  - Create dictionary
  - Join dictionary to tokens
  - Aggregate matches
  - Choose class
  - Refine dictionary
- Inductive Topic Classification

# **INDUCTIVE AND DEDUCTIVE**

# INDUCTIVE AND DEDUCTIVE

---

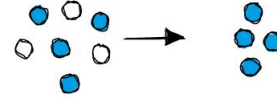


**PREPARATION:  
TOKENIZE TEXT**

# 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] ]
```

# DEDUCTIVE TOPIC CLASSIFICATION

## DEDUCTIVE TOPIC CLASSIFICATION APPROACH

---

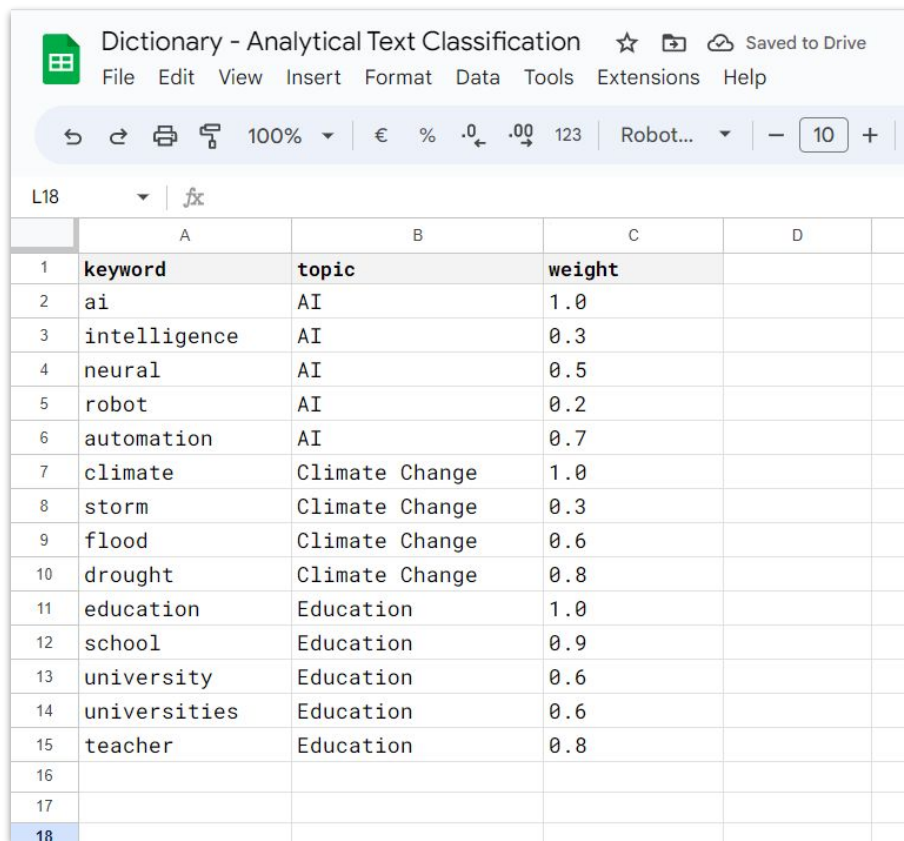
- Deductive text classification **starts with a predefined set of rules or hypotheses** and tests them against the data to classify it:
  - **Step 1:** Create a dictionary *keywords* → *topic* based on theory or hypotheses
  - **Step 2:** Join tokens with dictionary and extract keyword hits
  - **Step 3:** Aggregate keyword matches and assign topics. For instance:
    - Calculate absolute keyword count for each text (all and distinct hits)
    - Calculate relative keywords counts → Determine length of text first
    - Calculate weighted sum of keywords → Introduce weights to dictionary
  - **Step 4:** Choose class with highest score based on chosen metric
  - **Step 5:** Refine dictionary



# DEDUCTIVE TOPIC CLASSIFICATION

## STEP 1: CREATE DICTIONARY

- Driven by theory, we fill a dictionary in which we assign keywords to classes
- Classes can be hierarchical, e.g., *category* → *topic*
- Typically there exists a **1:n** relationship between class and keywords
- Optional: weights to distinguish important keywords from less important ones



Dictionary - Analytical Text Classification

File Edit View Insert Format Data Tools Extensions Help

100% | Robot... | 10

	A	B	C	D
1	<b>keyword</b>	<b>topic</b>	<b>weight</b>	
2	ai	AI	1.0	
3	intelligence	AI	0.3	
4	neural	AI	0.5	
5	robot	AI	0.2	
6	automation	AI	0.7	
7	climate	Climate Change	1.0	
8	storm	Climate Change	0.3	
9	flood	Climate Change	0.6	
10	drought	Climate Change	0.8	
11	education	Education	1.0	
12	school	Education	0.9	
13	university	Education	0.6	
14	universities	Education	0.6	
15	teacher	Education	0.8	
16				
17				
18				

## DEDUCTIVE TOPIC CLASSIFICATION

### STEP 2: JOIN DICTIONARY TO DATA

---

Load the dictionary (here: XLSX) and join with text data (here: transcripts from TED-talks). An inner join removes texts with no keyword hits (yet):

```
keywords <- read_excel("inductive_topics.xlsx")
```

```
ted_stop <-
```

*Resulting tibble from tokenization*

```
inner_join(keywords, by = join_by(word == keyword)) |>
```

## DEDUCTIVE TOPIC CLASSIFICATION

### STEP 3: COUNT MATCHES

---

Load the dictionary (here: XLSX) and join with text data (here: transcripts from TED-talks). A left join keeps the texts with no keyword matches (yet):

```
ted_stop |>
  inner_join(keywords, by = join_by(word == keyword)) |>
  group_by(title, topic) |>
  mutate(num_hits = n()) |>
  mutate(num_dist_hits = n_distinct(word))
```

## DEDUCTIVE TOPIC CLASSIFICATION

### STEP 3: CALCULATE RELATIVE COUNTS

---

Count the number of words per text before joining the dictionary and calculating the keywords hits:

```
ted_stop |>
```

```
group_by(title) |>  
mutate(num_words = n()) |>  
ungroup() |>
```

Determine the number of words per text

```
inner_join(keywords, by = join_by(word == keyword), keep = T) |>  
group_by(title, topic) |>  
mutate(num_distinct_hits = n_distinct(keyword)) |>  
mutate(num_total_hits = n()) |>
```

```
mutate(pct_total_hits = num_total_hits / num_words)
```

Use it to calculate relative count

## DEDUCTIVE TOPIC CLASSIFICATION

### STEP 3: WEIGHTED SUM

---

Introduce a weight column to the dictionary and apply it to calculate a weighted sum:

```
ted_stop |>
```

```
  inner_join(keywords, by = join_by(word == keyword), keep = T) |>
```

```
  group_by(title, topic) |>
```

```
  mutate(num_total_hits = n()) |>
```

```
  mutate(weighted_total_hits = sum(weight)) |>
```

```
  ungroup()
```

Sum up all weights for the keyword hits

## DEDUCTIVE TOPIC CLASSIFICATION

### STEP 4: CHOOSE CLASS

---

Choose the class with the highest score (here: relative counts):

```
ted_stop |>
  group_by(title) |>
  mutate(num_words = n()) |>
  ungroup() |>
  inner_join(keywords, by = join_by(word == keyword), keep = T) |>
  ...
```

```
distinct(id, topic, title, pct_total_hits) |>
```

Reduce to only one row per title/topic

```
group_by(id, title) |>
```

```
slice_max(order_by = pct_total_hits)
```

Get the top row within each id/title

## STEP 5: REFINE DICTIONARY

- 
- A word cloud featuring various terms related to climate change. The most prominent words are "climate" and "carbon" in large blue font. Other significant words include "change", "fossil", "global", "warming", "atmosphere", "planet", "years", "emissions", "trees", "solution", "energy", "soil", "time", "plan", "models", "earth", "amount", "natural", "greenhouse", "percent", "help", "countries", "business", "decarbonize", "people", "fuels", "dioxide", "clouds", "fuel", "country", "well", "human", "tax", "united", "soils". The words are arranged in a circular pattern, with some overlapping.

# INDUCTIVE TOPIC CLASSIFICATION



- Inductive text classification **starts with the data and extracts potential topics** into a dictionary while analyzing the data. The process is cyclic:
  - **Step 1:** Look at the most common tokens (words) that could be associated with a specific topic.
  - **Step 2:** Create a dictionary and add identified tokens and assign topics
  - **Step 3:** Apply dictionary to text data and refine:
    - A. Add new tokens for existing topics by looking at already classified texts
    - B. Identify new topics by looking at unclassified texts (or with low probability)