Text Mining and Sentiment Analysis

Prof. Annamaria Bianchi

A.Y. 2024/2025

Lecture 9

17 March 2025

Outline

Sentiment analysis (SA)

Dictionary-based approach

Polarity scoring

SA in the tidy data format

Packages: **tidytext, tidyverse**

Functions: `dplyr::inner_join(),tidyr::spread()`

Sentiment Analysis

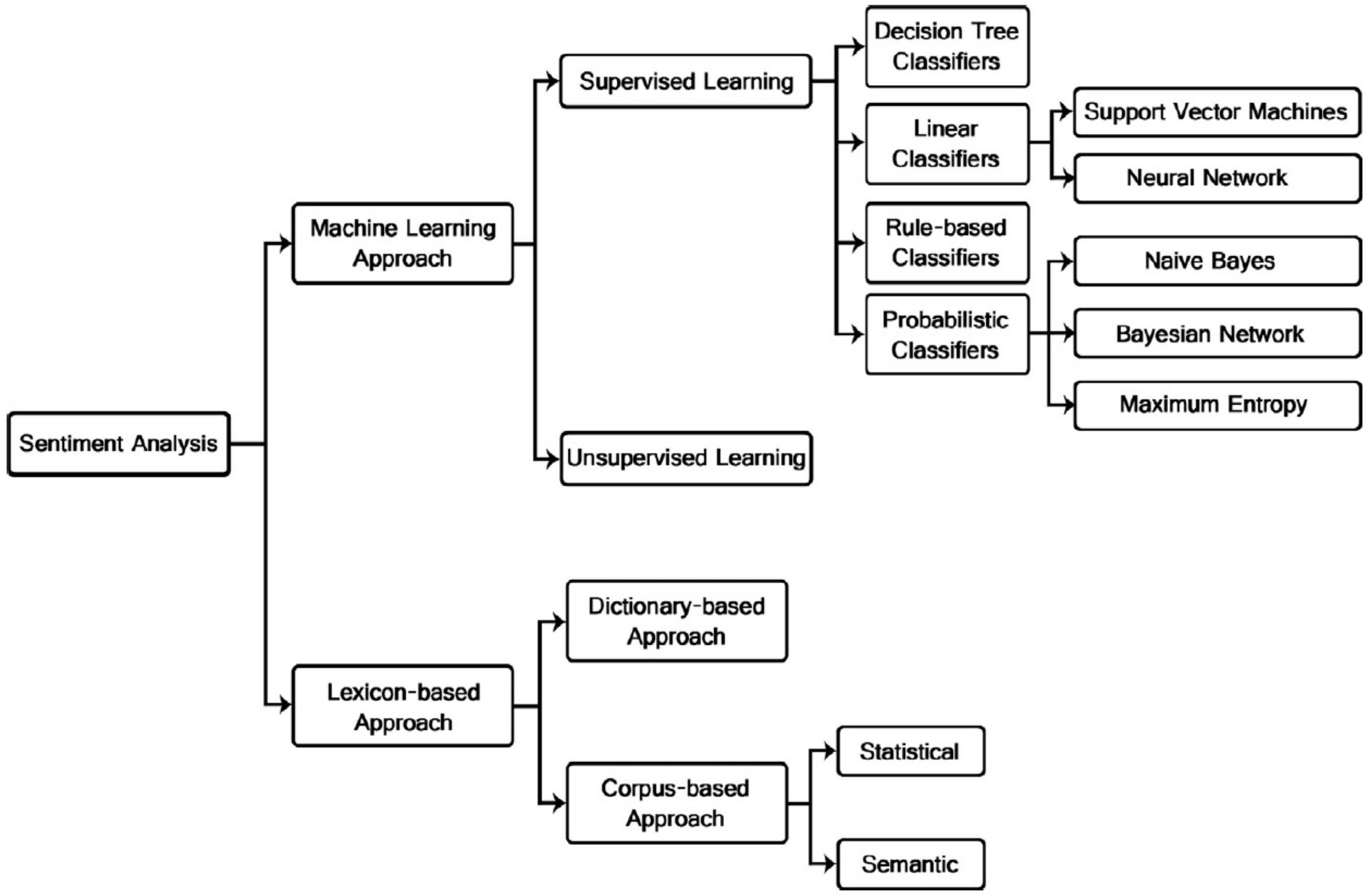
When human readers approach a text, we use our understanding of the emotional intent of words to infer whether a section of text is positive or negative, or perhaps characterised by some other more nuanced emotion like surprise or disgust. Sentiment analysis (also called opinion mining) uses the tools of text mining to approach the emotional content of text.

Sentiment analysis is the process of extracting an author's emotional intent from text.

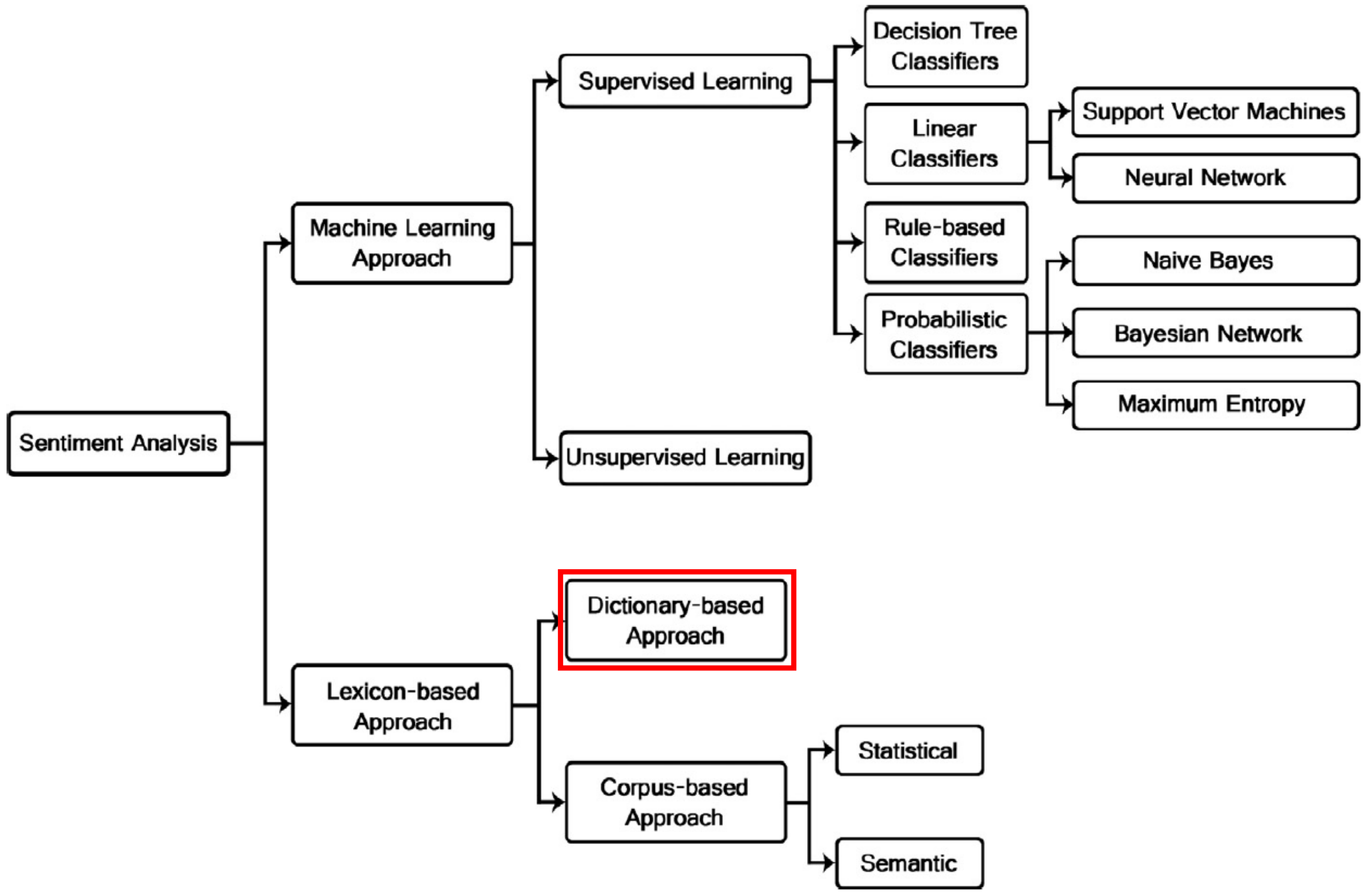
Sentiment analysis is very difficult to do well:

- It has interdisciplinary foundation, borrowing from disciplines such as linguistics, psychology and natural language processing
- Cultural and demographic differences between authors
- The human condition entails hundreds of related emotional states (happy *vs.* glad or bored *vs.* uninterested)
- Compounding sentiment analysis difficulties due to feature-specific sentiment

Sentiment Analysis



Sentiment Analysis



Sentiment Analysis

Lexicon-based approach: relies on a *sentiment lexicon*, a collection of known and precompiled sentiment terms. It involves calculating orientation for a document from the semantic orientation of words or phrases in the document. Dictionaries for lexicon-based approaches can be created manually or automatically, using seed words to expand the list of words.

Dictionary based approach: involves using a dictionary which contains synonyms and antonyms of a word. A simple technique in this approach is to use a small set of sentiment words (seeds) with known positive or negative orientations collected manually, then an algorithm grows this set by searching in any online available dictionary for their synonyms and antonyms.

Corpus based approach: helps to solve the problem of finding opinion words with context specific orientations. It identifies opinion words by considering word list. This approach begins with a seed list of opinion words, and then finds other opinion words in a large corpus to help finding opinion words with context specific orientation.

Statistical approach: co-occurrences of words are calculated to identify sentiment.
Semantic approach: terms are represented in semantic space to discover relation between terms → This principle assigns similar sentiment values to semantically close words.

Sentiment Analysis

Machine Learning approach (ML): applies ML algorithms to solve the SA as a regular text classification problem that makes use of syntactic and/or linguistic features.
Text Classification Problem Definition: it refers to the problem of assigning predefined classes or labels to unlabelled text documents. The classification model relates the features in the underlying record to one of the class labels.
Hard classification problem: when only one label is assigned to a record.
Soft classification problem: a probabilistic value of labels is assigned to a record.

Supervised learning technique: a set of training records, where each record is labeled to a class, is available. The labeled dataset is provided as input to train the model and this model is applied to test data to generate output. Several kinds of classifiers are available.

Unsupervised learning technique: used when it is difficult to find labelled training documents. No previous assumptions and definitions are given to the model about the outcome of variables feeded into it. Data (of course preprocessed before) are simply inserted, and one wants the model to learn the structure of the data itself.

Sentiment Analysis – dictionary-based approach

The dictionary-based approach considers the **text as a combination of its individual words**, and the sentiment content of the whole text as the **sum of the sentiment content of the individual words.**

This approach naturally takes advantage of the tidy tool framework.

This approach performs well with short and concise text but it has some drawbacks:

- Requires powerful linguistic resources that are not always available
- inability to find opinion words with domain and context specific orientations
- Urban slang and abbreviation
- Sarcasm
- It is based on unigrams
- Does not consider qualifiers before a word (e.g. «not good»)

Sentiment Analysis – polarity scoring

Beyond SA for emotional states, an easier approach is to merely state whether a document is positive or negative. This is referred to as **polarity** of a document.

The resulting polarity is a number that is negative to represent a negative, zero to represent neutral and positive to represent a positive tone.

Polarity can be easier than identifying the emotional state, as there are only two distinct classes.

Example. Surprise can be both positive or negative. Positive surprise may be «I just found out that I won the lottery», while negative surprise may be «I was just hit by a bus».
Rather than analyzing the nuanced differences of an emotional state like surprise, polarity of a document is often easier.

Sentiment Analysis – polarity scoring

SA in tidytext is based on the use of subjectivity lexicons. A subjectivity lexicon is a list of words associated with a particular emotional state.

Example. The words bad, awful, and terrible can be reasonably associated with a negative state. The words perfect and ideal can be associated with a positive state.

Given a lexicon, a simple approach consists in adding up the number of positive words in a document and subtracting the number of negative ones. The net result would yield a number and corresponding positive or negative tone of a passage.

Example. Consider the sentence

Sentiment analysis in R is good yet challenging.

Sentiment Analysis – polarity scoring

SA in tidytext is based on the use of subjectivity lexicons. A subjectivity lexicon is a list of words associated with a particular emotional state.

Example. The words bad, awful, and terrible can be reasonably associated with a negative state. The words perfect and ideal can be associated with a positive state.

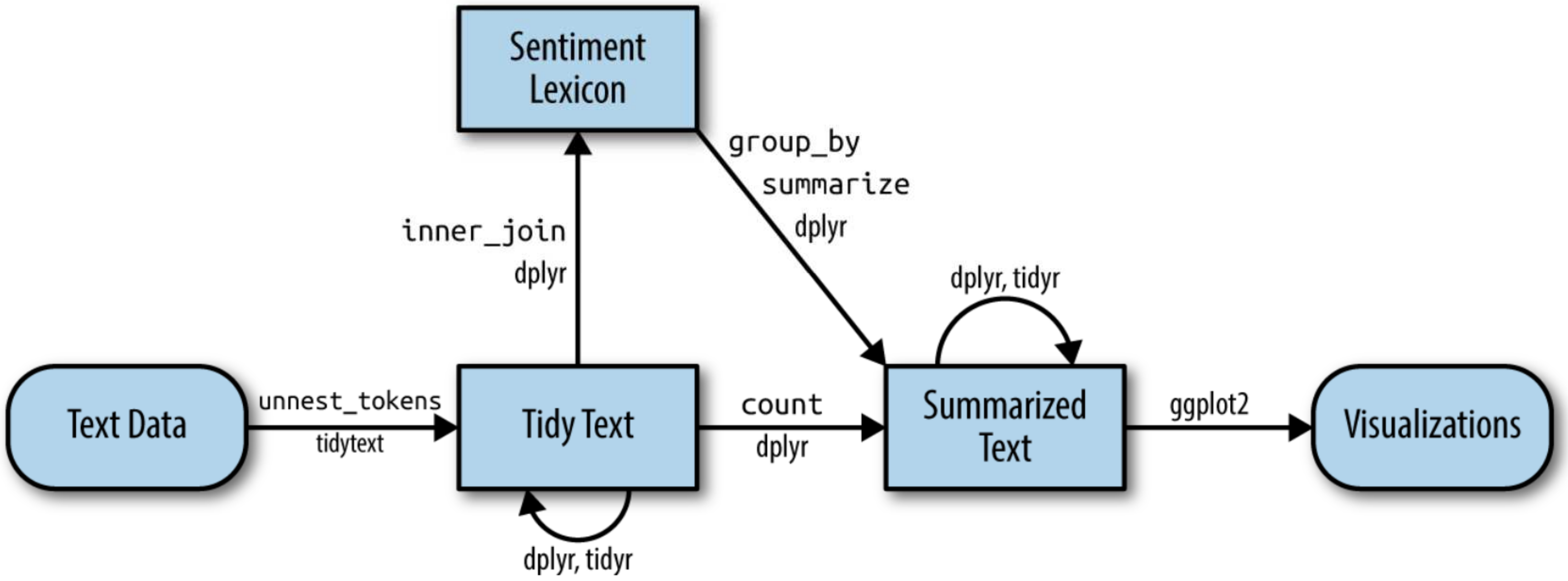
Given a lexicon, a simple approach consists in adding up the number of positive words in a document and subtracting the number of negative ones. The net result would yield a number and corresponding positive or negative tone of a passage.

Example. Consider the sentence

Sentiment analysis in R is $\overset{+}{\textit{good}}$ *yet* $\overset{-}{\textit{challenging.}}$

The word «good» has positive polarity, while the word «challenging» has negative polarity. The two cancel out, equalying zero. So the polarity of this sentence is zero.

Sentiment Analysis using the tidy data principle



Sentiment Analysis using the tidy data principle

We will perform SA using the **tidytext** package which contains several sentiment lexicons in the sentiment datasets.

For polarity calculations, tidytext uses a lexicon from research performed by Bing Liu (and collaborators) at the University of Illinois at Chicago. This lexicon is based on unigrams and contains approximately 6800 English words. Each word is assigned a positive or negative sentiment. Not every English word is in the lexicon because many English words are pretty neutral.

Sentiment Analysis using the tidy data principle

To get specific sentiment lexicon in a tidy format, with one row per word, we use the function `get_sentiments()`

```
get_sentiments(lexicon = c("bing", "afinn", "loughran", "nrc"))
```

`lexicon` The sentiment lexicon to retrieve; either "afinn", "bing", "nrc", or "loughran“

This function returns a **tbl_df** with a **word** column, and either a **sentiment** column (if lexicon is not "afinn") or a numeric value column (if lexicon is "afinn").

Sentiment Analysis using the tidy data principle

```
> library(tidyverse)
> library(tidytext)
> bing = get_sentiments("bing")
> bing
# A tibble: 6,786 × 2
  word        sentiment
  <chr>       <chr>
1 2-faces     negative
2 abnormal    negative
3 abolish     negative
4 abominable  negative
5 abominably  negative
6 abominate   negative
7 abomination negative
8 abort       negative
9 aborted     negative
10 aborts     negative
# … 6,776 more rows
```

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Scienze Economiche

Sentiment Analysis using the tidy data principle

Exercise. Explore the Bing lexicon.

a) How many positive and negative words does it contain, respectively?
b) Print the first 10 positive words and the first ten negative words.

Sentiment Analysis using the tidy data principle – case study

Suppose you have an apartment in Boston that you would like to rent through the Airbnb.com service. You want to make sure that your apartment has the qualities of a good rental.

Question: What quality properties are listed in positive or negative comments?

Data: After a stay, an Airbnb renter can leave comments about the property. These comments are public. You decide to analyze the comments for properties in Boston.
The dataset **bos_airbnb_1k.csv** contains 1000 randomly selected Boston airbnb listings.

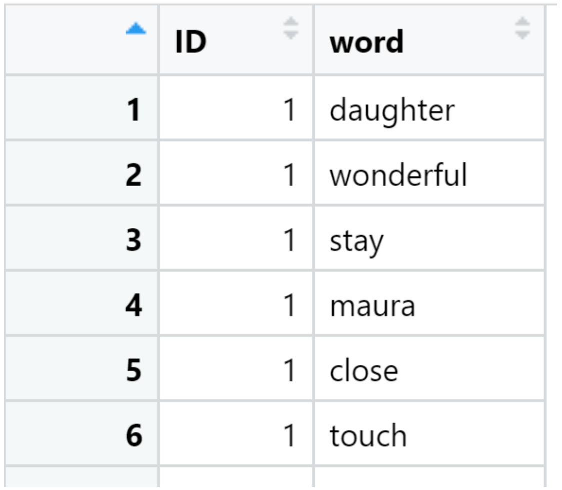
Open the dataset in the course *R* project.

```
> bos.airbnb = read_csv("bos_airbnb_1k.csv")
Rows: 1000 Columns: 91 ── Column specification
────────────────────────────────────────────────
Delimiter: ","
chr (54): date, reviewer_name, comments, listing_url, last_scraped,...
dbl (25): listing_id, reviewer_id, scrape_id, host_id, host_listing...
lgl (12): host_is_superhost, host_has_profile_pic, host_identity_ve...
```

Sentiment Analysis using the tidy data principle

Before applying SA, we select the variable comments, create an *id* variable for each comment, convert the data into the tidy format and drop stop words.

```
> bos.airbnb = bos.airbnb |>
+ select(comments) |>
+ mutate(ID = seq_along(comments))
> tidy.bos.airbnb = bos.airbnb |>
+ unnest_tokens(word, comments) |>
+ anti_join(stop_words)
Joining with `by = join_by(word)`
```



	ID	word
1	1	daughter
2	1	wonderful
3	1	stay
4	1	maura
5	1	close
6	1	touch

Sentiment Analysis using the tidy data principle

With data in a tidy format, SA can be done as an *inner join*, using the function **dplyr**::`inner_join()`. Given two data frames *x* and *y*, an inner join compares each row from *x* with each row from *y* and only keeps observations from *x* that have a matching key in *y*.

```
inner_join(x, y, by = NULL, ...)
```

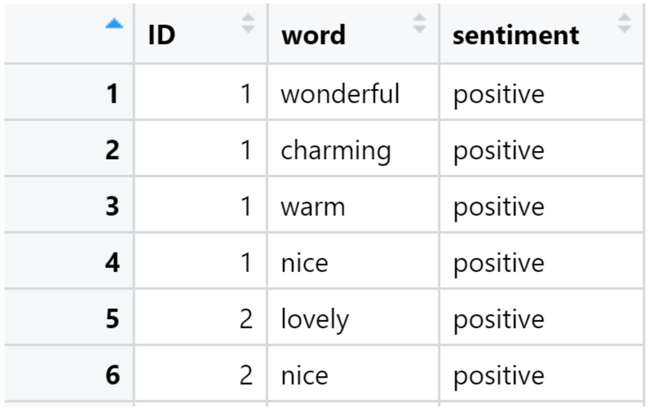
`x, y`	A pair of data frames, data frame extensions (e.g. a tibble)
`by`	A character vector of variables to join by. If `NULL`, the default, `inner_join()` will perform a natural join, using all variables in common across *x* and *y*. To join by different variables on x and y, use a named vector. For example, by = c("a" = "b") will match x$a to y$b.

The function adds columns from *y* to *x*, matching rows based on the keys. The function returns observations from *x* that have a matching key in *y*.

Sentiment Analysis using the tidy data principle

We use the inner join to identify the words in `tidy.bos.airbnb` that are contained in the bing lexicon. The rows with a match between *x* and *y* are returned.
Notice that, in the tokenization step, I chose the name `word` for the output column from `unnest_tokens()`. This is a convenient choice as the sentiment lexicons have column named word. Performing the inner join is thus easier.

```
> bos.pol = tidy.bos.airbnb |>
+ inner_join(bing)
Joining with `by = join_by(word)`
> View(bos.pol)
```



	ID	word	sentiment
1	1	wonderful	positive
2	1	charming	positive
3	1	warm	positive
4	1	nice	positive
5	2	lovely	positive
6	2	nice	positive

In case, I would like to specify the by variable

```
> bos.pol = tidy.bos.airbnb |>
+ inner_join(bing, by="word")
```

Exercise

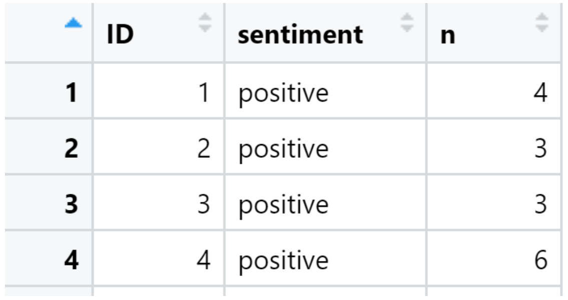
With reference to the Airbnb data on Boston apartments, write proper R code to perform the following steps

1. What is the total number of distinct positive words used in the comments? And the total number of distinct negative words?
2. What is the overall number of positive words used in the comments? And the overall number of negative words?

Sentiment Analysis using the tidy data principle

Next, use `count()` to summarise the observations

```
> bos.pol = bos.pol |>
+ count(ID, sentiment)
> View(bos.pol)
```



	ID	sentiment	n
1	1	positive	4
2	2	positive	3
3	3	positive	3
4	4	positive	6

For each comment, we obtain the number of positive and negative words

Sentiment Analysis using the tidy data principle

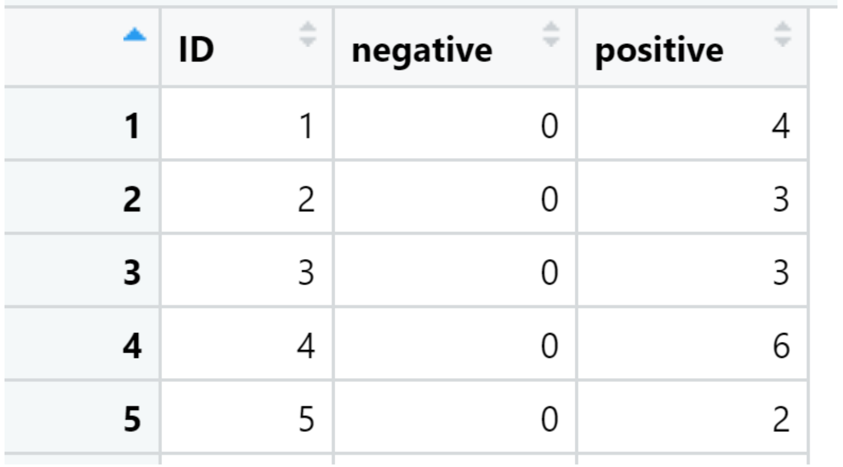
We then use the **tidyr**::`pivot_wider()` function to have negative and positive sentiment in separate columns

```
pivot_wider(data, names_from = name, values_from = value, values_fill = NULL,…)
```

`data`	A data frame to pivot.
`names_from`, `values_from`	A pair of arguments describing which column to get the name of the output column (`names_from`), and which column to get the cell values from (`values_from`).
`values_fill`	Optionally, a (scalar) value that specifies what each value should be filled in with when missing.

Sentiment Analysis using the tidy data principle

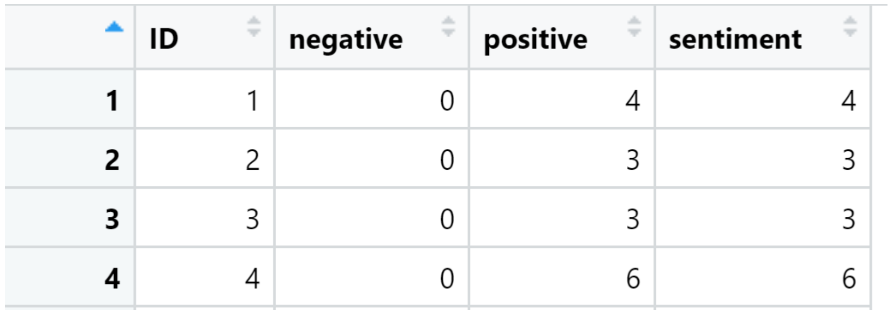
```
> bos.pol = bos.pol |>
+ pivot_wider(names_from = sentiment,
+ values_from = n,
+ values_fill = 0)
> View(bos.pol)
```



	ID	negative	positive
1	1	0	4
2	2	0	3
3	3	0	3
4	4	0	6
5	5	0	2

Finally, we compute the net sentiment (positive-negative)

```
> bos.pol = bos.pol |>
+ mutate(sentiment = positive-negative)
> View(bos.pol)
```



	ID	negative	positive	sentiment
1	1	0	4	4
2	2	0	3	3
3	3	0	3	3
4	4	0	6	6

Sentiment Analysis using the tidy data principle

Generally you apply all the steps together:

```
> bos.pol = tidy.bos.airbnb |>
+ inner_join(bing, by="word") |>
+ count(ID, sentiment) |>
+ pivot_wider(names_from = sentiment,
+ values_from = n,
+ values_fill = 0) |>
+ mutate(sentiment = positive-negative)
```

Exercises for you

Exercise 1. With reference to the Airbnb data on Boston apartments, compute summary statistics for the polarity score.

Exercise 2. Plot the distribution of the polarity scores using an histogram