

Part II-1: Scraping, Text Normalization & Static Feature Extraction



Outline

- i. Scraping
 - i. General Web Scraping
 - ii. Scraping Twitter
- ii. Text Normalization
 - i. Regular Expressions
 - ii. Stemming, Lemmatization
- iii. Static Feature Extraction

Part II-1: Scraping, Text Normalization & Static Feature Extraction

Scraping

Scraping Idea

- **Goal:** access, use & analyze data available on the internet
- **Problem:** unstructured, HTML-formatted, non-downloadable data


LEARN HOW TO WEB SCRAPE

Example 1

To keep things simple to begin with lets try to just extract the three titles, associated text and images for each of the 3 news stories.

After running and understand the script why not have ago at creating your own web scraper for [Exercise 1](#).

New
Neighbourhood watch.



SITE MAP

- Home
- Example 1
- Example 2
- Example 3
- Example 4
- Exercise 1
- Exercise 2
- Exercise 3
- Resources



```
<!DOCTYPE html>
<html lang="en-GB">
  <link type="text/css" rel="stylesheet" id="dark-mode-custom-link">
  <link type="text/css" rel="stylesheet" id="dark-mode-general-link">
  <style lang="en" type="text/css" id="dark-mode-custom-style"></style>
  <style lang="en" type="text/css" id="dark-mode-native-style"></style>
  <head></head>
  <body class="page-template-default page page-id-25 custom-background wp-embed-responsive customizer-styles-app
    lied has-marketing-bar highlander-enabled highlander-light vsc-initialized" style="">
    <div id="page" class="site"></div>
    <!-- #page -->
    <!-- -->
    <div id="marketingbar" class="marketing-bar noskim"></div>
    <script src="//0.gravatar.com/js/gprofiles.js?ver=202113y" id="profiles-cards-js"></script>
    <script id="wpgroho-js-extra">
      var WPGroho = {"my_hash":""};
    </script>
    <script type="text/javascript" src="https://s1.wp.com/wp-content/mu-plugins/gravatar-hovercards/wpgroho.js?m
      a1610357208"></script>
    <script></script>
    <div style="display:none">
      </div>
    <!-- CCPA [start] -->
    <script type="text/javascript"></script>
    <!-- CCPA [end] -->
    <script type="text/javascript"></script>
    <script></script>
    <div id="carousel-reblog-box"></div>
    <div class="jp-carousel-wrap jp-carousel-transitions" itemscope itemtype="https://schema.org/ImageGallery"
      style="display: none;"></div>
    <script type="text/javascript"></script>
    <link rel="stylesheet" id="all-css-0-2" href="https://s1.wp.com/wp-content/mu-plugins/carousel/jetpack-carou
      sel
    <script
      var c
```

<https://practicewebscrapingsite.wordpress.com/example-1/>

Scraping Learning to Scrape

- Sorry, but: it's a tedious thing
 - Highly manual, time-consuming process
 - Need for adjustment every time the website source code changes
- Harder with more complex websites optimized for UX
- Examples
 - Text data from Wikipedia, labeled image data from Google
 - Social media data from Twitter, Facebook, ...
 - Reviews, feedbacks from Amazon



<https://practicewebscrapingsite.wordpress.com/>

Scraping Steps

- **Basic steps**

1. Parse (static) website content
2. Grasp page set-up



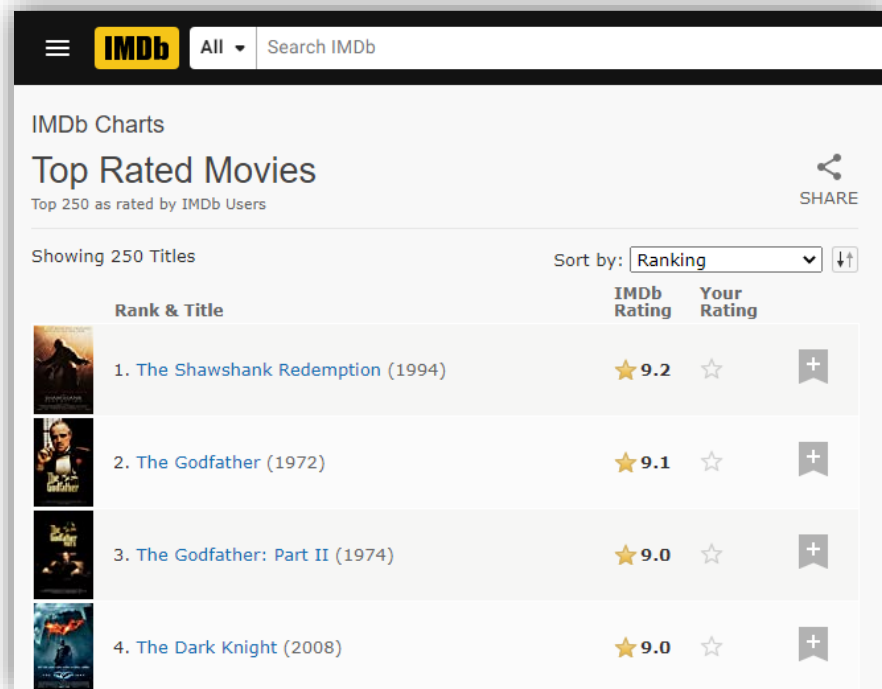
requiring CSS selectors – easier with <https://selectorgadget.com/>

3. Extract information

- **Advanced:** actual website navigation, i.e., clicking buttons and jumping to pages, with a remotely controlled browser

Scraping Example

- **Goal:** get TOP 250 movie rankings and ratings from IMDb
http://www.imdb.com/chart/top?ref=mv_250_6



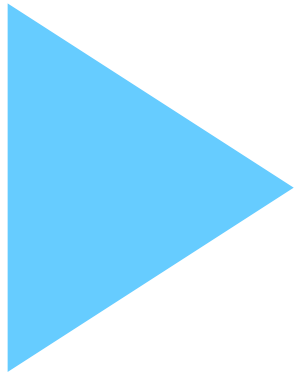
##	ranking	title	year	rating
## 1:	1	Die Verurteilten	1994	9.2
## 2:	2	Der Pate	1972	9.1
## 3:	3	Der Pate 2	1974	9.0
## 4:	4	The Dark Knight	2008	9.0
## 5:	5	Die zwölf Geschworenen	1957	8.9
## 6:	6	Schindlers Liste	1993	8.9

Scraping Example

- **Guide**

1. Install and load package rvest:
`install.packages("rvest"); library(rvest)`
2. Parse webpage: `read_html()`
3. Explore contents using Chrome's **developer tab** (clicking F12) or some helper tool, e.g., **SelectorGadget** (does not require technical knowledge of HTML and CSS)
4. Extract information from the webpage: `html_nodes()`
5. Store extracted information in desired format: `html_text()`

Scraping Example



Demo 1: Web Scraping Example

Scraping Twitter Data

- **Guide**

1. Install and load package `rtweet`

2. Set up **Twitter API**

1. Create Twitter account: <http://twitter.com/signup>

2. Apply for a developer account by filling out a short application form:
<https://developer.twitter.com/en/apply-for-access.html>

3. Click on „key and access token“ and get your API access:
consumer key (API key), consumer secret (API secret),
access token, access token secret



keep somewhere safe

3. Use the keys as arguments: `rtweet::create_token(consumer_key, consumer_secret, access_token, access_secret)`

Scraping Twitter Data – Limitations

- Using the **standard** (free) search API
 - Tweets only for a 6-9 days period (for more information see <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html>)
 - Scraping of up to 18,000 tweets possible
 - Package documentation: <https://cran.r-project.org/web/packages/rtweet/rtweet.pdf>



useful tutorial on <https://rtweet-workshop.mikewk.com/>

- Worth considering: scraping in Python using BeautifulSoup

Scraping Twitter Data – Example

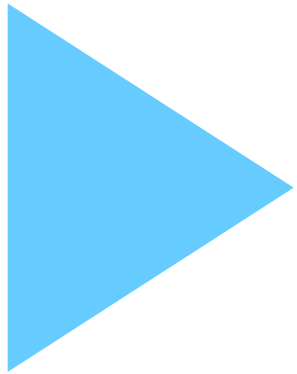
- **Goal:** scrape tweets and associated information



Screenshot from: <https://twitter.com/JKasek/status/1377285533274083343>

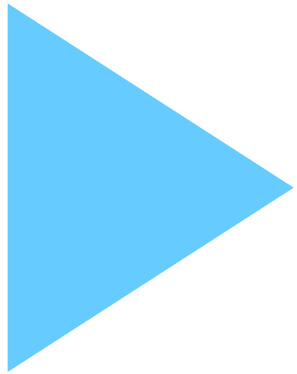
	created_at	screen_name	text	retweet_count	status_id	favourites_count
1	2021-03-31 15:43:54	JKasek	#Sachsen: die #CDU stimmt in #Plauen gemeinsam mit der #AF...	638	1377285533274083343	1731

Scraping Twitter Data



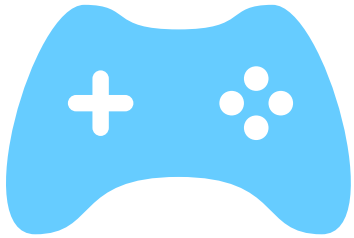
Demo 2: Twitter Scraping Example

Scraping Twitter Data



Demo 3: Twitter Scraping Python

Scraping Exercise



Exercise 1: Scraping

Part II-1: Scraping, Text Normalization & Static Feature Extraction

Text Normalization

Text Normalization Purpose

- Series of steps to clean and standardize textual data
- **Goal:** representation of texts by meaningful tokens that co-occur across documents as much as possible
- **Basic techniques:**
 - Removing stopwords: words with little or no significance
 - Removing special characters (symbols, punctuation, HTML entities etc.)
 - Stemming, lemmatization

Die Ausgrenzung von MigrantInnen von der #EssenerTafel ist inakzeptabel und rassistisch. Wir dürfen nicht zulassen, dass die Ärmsten gegeneinander ausgespielt werden.

Text Normalization **Regular Expressions**

- Focus of NLP in general: analysis and understanding of (unstructured) text
- **Regular expression (regex):** pattern (= sequence of characters) defined to search text with a common structure
- Used for
 - searching for a specific file name,
 - finding a text with a specific pattern,
 - replacing a specific pattern in a text, etc.
- Standardized across many programming languages

Text Normalization Regular Expressions

- `stringr`: useful R package to deal with all kinds of text wrangling
- Important commands in `base` & `stringr`:

	base	stringr
Identify	<code>grep(., value = FALSE)</code>	<code>str_detect()</code>
Extract	<code>grep(., value = TRUE)</code>	<code>str_extract()</code>
Locate	<code>gregexpr()</code>	<code>str_locate()</code>
Replace	<code>gsub()</code>	<code>str_replace()</code>
Split	<code>strsplit()</code>	<code>str_split()</code>

Text Normalization Useful Regex Patterns

Pattern	Function
<code>\d</code> or <code>[:digit:]</code> or <code>[0-9]</code>	Matches any digit
<code>[:alpha:]</code> or <code>[A-Za-z]</code>	Matches any character
<code>[a-z]</code> or <code>[:lower:]</code>	Matches any lowercase character
<code>[A-Z]</code> or <code>[:upper:]</code>	Matches any uppercase character
<code>[abc]</code>	Matches a, b or c
<code>[^abc]</code>	Matches anything except a, b, or c.
<code>[:punct:]</code>	Matches punctuation characters: ! " # \$ % & ' () * + , - . / : ; < = > ? @ [] ^ _ ` { } ~
<code>a(b c)d</code>	Matches abd or acd
<code>^x</code>	Matches x if string begins with x
<code>x\$</code>	Matches x if string ends with x

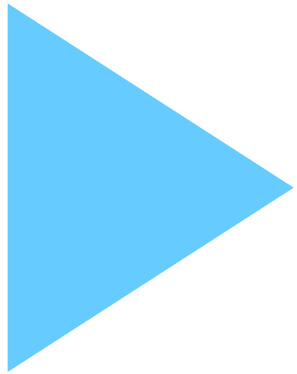
Text Normalization Useful Regex Patterns

Pattern	Function
<code>x?</code>	Matches 0 or 1 occurrences of <code>x</code>
<code>x*</code>	Matches 0 or more occurrences of <code>x</code>
<code>x+</code>	Matches 1 or more occurrences of <code>x</code>
<code>x{n}</code>	Matches exactly <i>n</i> occurrences of <code>x</code>
<code>x{n,}</code>	Matches <i>n</i> or more occurrences of <code>x</code>
<code>x{,m}</code>	Matches at most <i>m</i> occurrences of <code>x</code>
<code>x{n,m}</code>	Matches between <i>n</i> and <i>m</i> occurrences of <code>x</code>
<code>x(?:=...)</code>	Matches <code>x</code> if followed by ... (for negation, replace = by !)
<code>(?:<=...)x</code>	Matches <code>x</code> if preceded by ... (for negation, replace = by !)



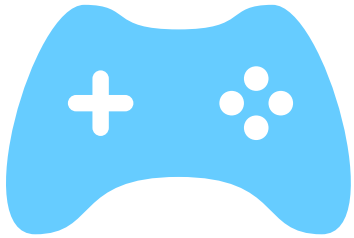
<https://cran.r-project.org/web/packages/stringr/vignettes/regular-expressions.html>

Text Normalization Regular Expressions



Demo 4: Regular Expressions

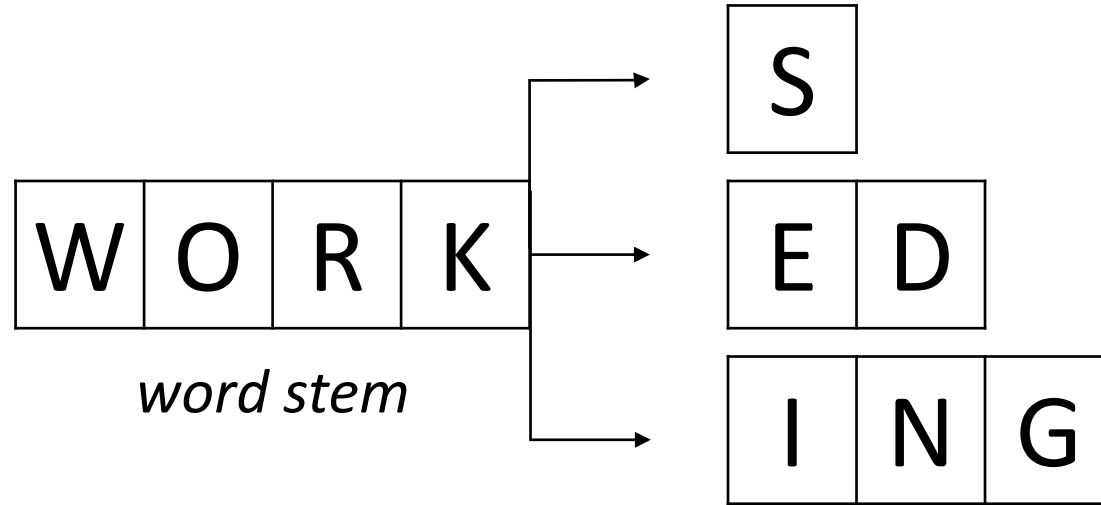
Text Normalization Exercise



Exercise 2: Regular Expressions

Text Normalization Stemming

- **Idea:** retrieve base form, the root stem



- Example in German: Bruder – Bruders – brüderlich/e/n/r/s – Brüderlichkeit/en → **bruder**

Text Normalization Lemmatization

- **Problem with stemming**

- Potentially erroneous
- Overstemming: *politics* → *polit*
- Understemming: *travels* → *trav* but *travelled* → *travel*

- Alternative: **lemmatization**

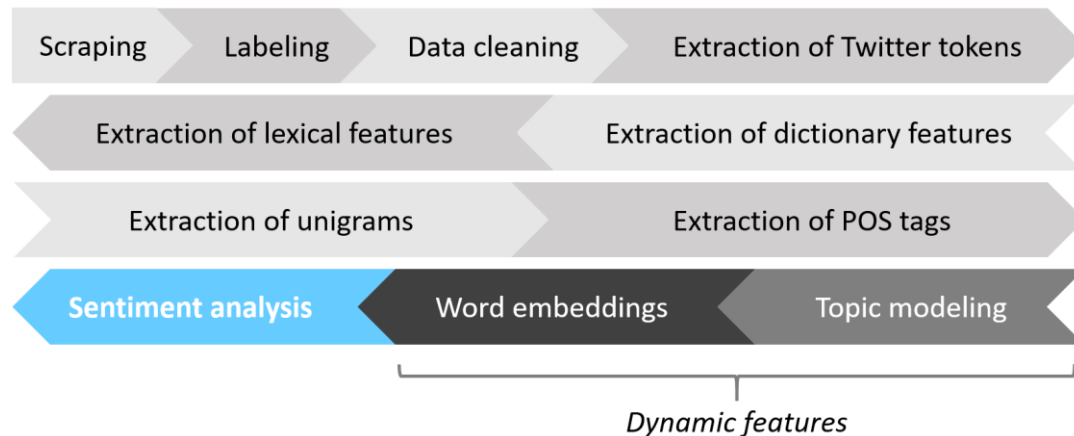
- Retrieving the root **word** (not root stem)
- Difference: the lemma will always be present in the dictionary (lexicographically correct word)
- Slower than stemming
- Potentially more difficult for grammatically complex languages

Part II-1: Scraping, Text Normalization & Static Feature Extraction

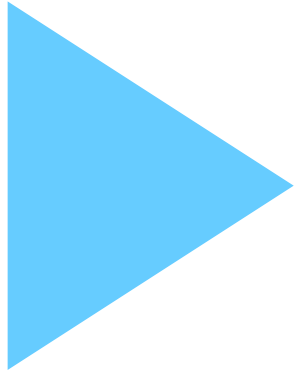
Static Feature Extraction

Static Feature Extraction Basic Text Cleaning

- **Now, back at our task**
- We need to
 - Perform basic text cleaning
 - Extract all static features we wish to use for sentiment classification



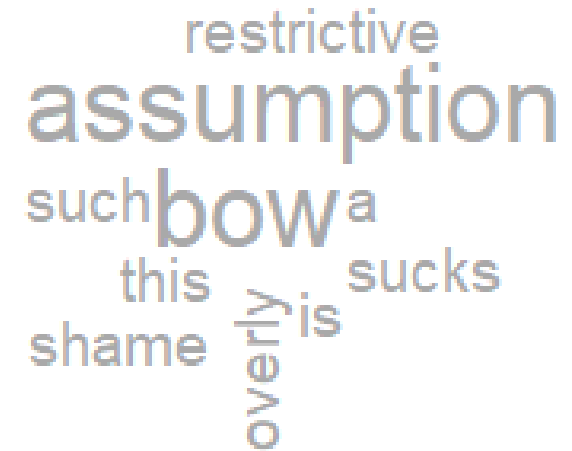
Static Feature Extraction Basic Text Cleaning



Demo 5: Basic Text Cleaning

Static Feature Extraction Features

- One **central assumption** in sentiment analysis with standard machine learning techniques: **bag-of-words (BOW)**
- Why?
 - Standard machine learning only built for tabular data
 - Consequence: discarding all information about word order, sentence structure, ...
 - **Eye-watering simplification**, but it is what it is



restrictive
assumption
such bow a
this is sucks
shame overly

Static Feature Extraction Features

- **Useful (static) features in sentiment analysis**
 - Polarity clues
 - Negations, intensifications, punctuations, repetitions
 - Word/character n -grams
 - Part-of-speech (POS) tags
 - Twitter-specific features
- **Recall our goal:** numeric representation of texts by tokens that co-occur across documents

Static Feature Extraction Features

- **Polarity clues**

- **Idea:** find sentiment-bearing tokens
- **Details**
 - Presence/absence or count
 - Positive/negative, positive/negative/neutral, more fine-grained emotions, ...
- **Computation:** look-up using publicly available dictionaries



if useful: modify/enrich

What a **despicable** thing to do, I **hate** him! → *positive: 0, negative: 2*

Static Feature Extraction Features

- **Negations, intensifications, punctuations, repetitions**
 - **Idea:** capture meaning modifiers (lost with BOW assumption!)
 - **Assumptions**
 - Negations might flip sentiments.
 - Intensifications might indicate/strengthen sentiments.
 - Punctuations/repetitions might indicate sentiments.
 - **Computation:** look-up using regular expressions

I **cannot** recommend this movie.....

A **truly** grand and **deeply** moving plot.

Static Feature Extraction Features

- **Word/character n -grams**
 - **Idea:** count general tokens to represent texts
 - **Details**
 - n -gram: sequence of n words/characters – somewhat mitigating BOW effect
 - Unigrams, bigrams, trigrams, ...
 - The larger n , the lower the probability of n -grams occurring in multiple documents
 - **Computation:** count using available functionalities (e.g., in quanteda)

Bello the dog is a good boy.

bello	dog	good	boy	a	b	d	e	g	h	i	l	o	s	t	y
1	1	1	1	1	2	2	2	2	1	1	2	5	1	1	1

Static Feature Extraction Features

- **POS tags**

- **Idea:** capture grammatical structure

<https://universaldependencies.org/u/pos/all.html>

- **Details**

- Computed on full text
 - Assign each word a grammatical role (18 universal tags)



- **Assumption:** presence of many adverbs/adjectives indicative of sentiments
 - **Computation:** use available parsers (e.g., in spacyR)

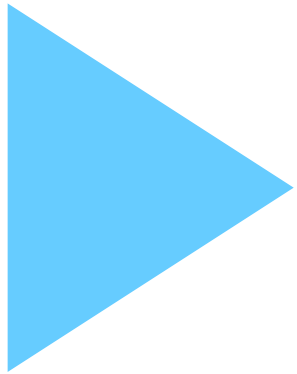
Bello	the	dog	is	a	good	boy	.
PROPN	DET	NOUN	AUX	DET	ADJ	NOUN	PUNCT

Static Feature Extraction Features

- **Twitter-specific features**
 - **Idea:** exploit Twitter-inherent tokens
 - **Details**
 - Emojis: count/assign polarity
 - Hashtags: count/mine (for topics, meaning, ...)
 - Tags: count/mine
 - ...
 - **Computation:** look-up using regular expressions

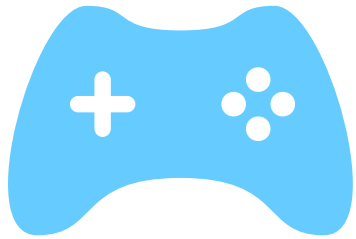
I love doing dishes #not 😬

Static Feature Extraction Features



Demo 6: Static Feature Extraction

Static Feature Extraction Exercise



Exercise 3: Static Feature Extraction

Part II-1: Scraping, Text Normalization & Static Feature Extraction

Literature and References

Berry, M., and Kogan, J. (2010): Text Mining. Applications and Theory, John Wiley & Sons, Inc.

Kearney, M., Heiss, A., and Briatte, F. (2020): rtweet. Collecting Twitter Data, R package version 0.7.0, URL: <https://CRAN.R-project.org/package=rtweet>

Miner, G., Elder IV, J., Fast, A., Hill, T., Nisbet, R. and Delen, D. (2012): Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications, Academic Press.

Munzert, S., Rubba, C., Meißner, P., and Nyhuis, D. (2015): Automated Data Collection with R : A Practical Guide to Web Scraping and Text Mining, John Wiley & Sons, Inc.

Silge, J. (2017): Text Mining with R. A Tidy Approach, O'Reilly.

Wickham, H (2019): stringr. Simple, Consistent Wrappers for Common String Operations, R package version 1.4.0, URL: <https://CRAN.R-project.org/package=stringr>

Wickham, H (2021): rvest. Easily Harvest (Scrape) Web Pages, R package version 1.0.0, URL: <https://CRAN.R-project.org/package=rvest>