# Documentation for dataset: agency-data

*Last edited: 2020-02-11*

GitHub repo: https://github.com/CALDISS-AAU/aiframing (currently on branch raw_update)

## About the dataset

The dataset consists of scraped articles of various consultant agencies. The current version contains articles from the following consultant agencies:

- KPMG
- EY

A .csv and .feather version of the dataset is stored.

## About the pre-processing and tokenizing

The articles are pre-processed and tokenized using the script: json_to_df.R (located in the preprocessing folder on the GitHub repository).

The script compiles a dataset consisting of the raw texts, metadata as well as various pre-processed and tokenized versions of the texts (see the 'Variables' section).

*NOTE on the tokenization ( `tokens` variable):* In order to keep the most common bigrams as individual tokens ('artificial intelligence', 'machine learning', 'data analytics'), part of the script looks for the 50 most common bigrams (based on count) and keeps these as tokens in the `tokens` variable instead of the individual words. The code is rather crude and does in its current state also keep some bigrams that are not meaningful as bigrams. This is expected to be corrected with the addition of more texts.

## Datacollection

The data is collected via individually build scrapers for each consultant agency website. Setting up the comparable parameters for collection is difficult as the different consultant agencies uses different forms of articles.

The scrapers are build to gather the texts (articles, blogposts, briefs and the like) which are themed around "artificial intelligence"

## EY scraper

The EY scraper starts at EY's subpage about artificial intelligence: https://www.ey.com/en_gl/ai

The URL's for the individual articles on the site are gathered and the individual articles parsed. The scraper is built to scrape the material related to the actual content of the article while avoiding elements related to the general structure of the site.

## KPMG scraper

The KPMG scraper starts at KPMG's subpage about 'Data-driven technologies', as there is not a specific subpage on artificial intelligence: https://home.kpmg/xx/en/home/insights/2018/08/data-driven-technologies.html (NOTE: This URL is no longer valid)

The URL's for the individual articles ("insights") are gathered and the individual articles parsed. The scraper is built to scrape the material related to the actual content of the article while avoiding elements related to the general structure of the site.

## McKinsey scraper

The McKinsey scraper starts at a URL sending a search query for "artificial intelligence": https://www.mckinsey.com/search?q=artificial+intelligence

The URL's for the individual results are gathered and the individual pages parsed. The pages contain various contents: articles, blog post, podcast or video transcriptions, report excerpts, profiles etc. All content types except for profiles are scraped. Information on the content type is stored in the variable `text_type`.

The scraper is built to scrape the material related to the actual content of the article while avoiding elements related to the general structure of the site.

## Bain & Company scraper

The Bain & Company scraper starts at a URL sending a search query for "artificial intelligence" filtering for content types "articles" and "briefs": https://www.bain.com/search/?searchValue=artificial+intelligence&filters=%7Ctypes%28426%2C427%29&pageNumber=0&sortValue=date

The URL's for the individual results are gathered and the individual pages parsed. Information on the content type is stored in the variable `text_type`.

The scraper is built to scrape the material related to the actual content of the article while avoiding elements related to the general structure of the site.

# Variables

The dataset contains the following variables:

- url
- title
- article.date
- modified.date
- download.date
- text
- text.type
- agency
- text_clean

- text_pp
- tokens_raw
- tokens
- text_length

**url**

Contains the URL of the scraped text.

**title**

Contains the title of the text.

**article.date**

Contains the date the article was published.

**modified.date**

Contains the date the article was last modified (if available).

**download.date**

Contains the date the text was last scraped.

**text**

The raw text of the article.

**text.type**

Contains information about the type of text as labelled by the consultancy agency (if available).

**agency**

The consultant agency that published the article.

**text_clean**

The texts cleaned of "filler" text and stripped of whitespace (newline characters, tabulate characters etc.).

Filler text is understood as standard text elements that appear in almost all texts from a specific consultant agency. This includes mission statements, read more sections, share on social media functions and so on.

**text_pp**

Contains a pre-processed, non-tokenized version of the text (from `text_clean`).

The text from `text_clean` is preprocessed in the following steps (in order):

1. Punctuation removed
2. Numbers removed
3. Lowercase words containing less than 3 characters removed
4. Uppercase words containing 1 character removed
5. Text converted to lowercase
6. Stopwords removed (standard list for english stopwords from the R `tm` package is used)

**tokens_raw**

List of tokens in the text based on the text in `text_clean`.

The text is tokenized in the following steps (in order):

1. Punctuation removed
2. Numbers removed
3. Tokenized using `Boost_tokenizer` (from the R `tm` package)

**tokens**

List of tokens in the text based on the text in `text_pp`.

The text is tokenized in the following steps in addition to the steps taken in creating `text_pp` (in order):

1. Tokenized using `Boost_tokenizer` (from the R `tm` package)
2. Most common bigrams (top 50 based on count) are inserted as bigrams tokens (replacing the individual tokens in those bigrams)