SMA: Task I

Use the dataset `tagesschau_articles.json` for exercises 2, 3, and 4.

1 Exercise 1: Web Scraping (25 points)

Write a function that scrapes articles from https://www.tagesschau.de/archiv/. In particular, the function shall take a date as input, and return a Pandas DataFrame of all the articles published by Tagesschau on this day. Organize the DataFrame such that a single row represents one article, and that the columns represent the following items: title, date, short summary text (in bold), regular text, tags.

Demonstrate that your function works for the day '2022-11-01'. Print the shape of your data, and display the first five rows.

2 Exercise 2: Word Cloud (25 points)

Write a function that is able to create two types of wordclouds for a given Tagesschau article: (1) a wordcloud of nouns, and (2) a wordcloud of named entities (may include all kinds of named entities, such as persons, organizations, geographic entities, etc.). As is typical for wordclouds, the number of occurrences of a given noun or entity should be represented through its size in the wordcloud. The function shall have at least the following 4 arguments:

- `id`: article id
- `type`: 'noun' or 'entity'
- `width`: width of the wordcloud
- `height`: height of the wordcloud.

Demonstrate that your function works using `wordcloud(id=11, type='noun')` and `wordcloud(id=11, type='entities')`

3 Exercise 3: Train word embeddings (25 points)

In this exercise, the overall task is to train word embeddings based on the Tagesschau corpus. Hint: the Gensim package offers a performant implementation of the relevant models. In particular, you can speed up the process by running the training in parallel on multiple cores.

- Before the training, normalize the corpus by removing German stopwords. Analyze whether further cleaning steps are necessary, and if so carry out these steps. But to keep the subsequent analysis consistent, do NOT stem or lemmatize words.
- Train word embeddings using (1) word2vec (CBOW model), and (2) fastText. For both models,
  - set the dimension of the word vectors to 80
  - set the size of the context window to 6
  - ignore all words occuring less than 6 times in the entire corpus
  - set all other arguments at your own discretion, or take the defaults. But make sure that the settings are comparable for CBOW and fastText.

4 Analyse and apply word embeddings (25 points)

- What are the 10 most similar words to "König" according to (1) word2vec and (2) fastText, respectively?
- Carry out the same type of comparison for two or more other words of your choice. Provide at least one example, where you consider word2vec's recommendations better, and one example where you prefer fastText's recommendation.
- Describe in what sense the results differ or look comparable. Explain your findings by relating them to the conceptual differences/communalities of word2vec and fastText. Based on this small-scale analysis, which of the two models would you prefer for downstream machine learning tasks. Justify your answer.
- Create a content-based recommender for Tagesschau articles. Specifically, write a `recommend` function that takes the `id` of an article as an input, and returns the 5 most similar articles (you may return the five rows from the original DataFrame). Use your preferred embedding model from above to compute the article similarities. Demonstrate that your function works using `recommend(id=11)`.