Part 3: Ranking & Filtering

In this part of the project, you will experiment with different ranking algorithms that can be applied in a search engine. Your task is to design and implement a retrieval pipeline that:

- Takes a query as input (a piece of text).
- Finds all documents that contain all query terms (conjunctive query, i.e., AND semantics).
- Sorts the matching documents by relevance using different ranking methods.

The main goal of this assignment is to explore and compare various relevance scoring approaches. By the end, you should be able to analyze how different algorithms affect the ranking order of documents.

Important: For this assignment, we only consider conjunctive queries (AND). This means that a document is included in the results only if it contains every word from the query.

SCORE:

1. You're asked to provide 3 different ways of ranking:
    a. **TF-IDF + cosine similarity**: Classical scoring, which we have also seen during the practical labs
    b. **BM25**
    c. **Your Score:** Here, the task is to create a new score. (Be creative 🎨, think about what factors could make a document more relevant to a query and include them in your formula.)

    Explain how the ranking differs when using TF-IDF and BM25, and think about the pros and cons of using each of them. Regarding your own score, justify the choice of the score (pros and cons). HINT: Look into numerical fields that each record has to build your score.

2. Implement **word2vec + cosine** ranking score. Return a top-20 list of documents for each of the 5 queries defined in the Part 2 of your project, using search and word2vec + cosine similarity ranking.

    To represent a piece of **text** using **word2vec**, we create a **single vector** that represents the entire text. This vector has the same number of dimensions as the word vectors and is calculated by **averaging the vectors of all words** in the text.

Example:
Consider the text:

"Wireless Bluetooth headphones with noise cancellation"

Suppose we have Word2Vec vectors for each word:

- Wireless → v1
- Bluetooth → v2
- headphones → v3
- with → v4
- noise → v5
- cancellation → v6

All vectors have the same number of dimensions. To represent the text as a single vector, we **average the word vectors**:

Text vector = (v1 + v2 + v3 + v4 + v5 + v6) ÷ 6

The resulting vector has the same number of dimensions as the individual word vectors and represents the content of the entire text. This approach allows us to compare texts based on their vector representations for tasks like search or recommendation.

3. Can you imagine a better representation than word2vec? Justify your answer.
   (**HINT** - what about Doc2vec? Sentence2vec? What are the pros and cons?)