**Document Similarity Checker: In-Depth Project Workflow**

This tool automates the process of identifying similar documents within a dataset using advanced NLP and machine learning routines. The project aims to provide useful insights for tasks like clustering, duplicate detection, and information retrieval.

**1. Data Structure and Organization**

* The repository is structured with explicit folders for source code (src/), raw and processed data (data/raw, data/processed), output visualizations (outputs/), exploratory notebooks (notebooks/), and tests (tests/). Each Python module addresses a key part of the workflow:
  + main.py: Master script orchestrating the workflow.
  + similarity.py: Contains functions for vectorizing documents and calculating cosine similarity.
  + preprocessing.py: Implements NLP steps such as tokenization, stopword removal, and lemmatization.
  + visualization.py: Handles heatmap generation for similarity scores.

**2. Data Loading and Preprocessing**

* The project begins by loading a CSV dataset of documents. Preprocessing transforms each document to a clean, standardized text form:
  + **Tokenization**: Breaks text into individual words or tokens, a prerequisite for vectorization.
  + **Stopword Removal**: Filters out tokens like "the", "is", "and", which generally carry little semantic weight.
  + **Lemmatization**: Converts words to their base or root form (e.g., "running" → "run") using NLTK or other tools, ensuring consistent word representations.

**3. Feature Extraction: TF-IDF Vectorization**

* Texts are converted into TF-IDF vectors, where each entry captures:
  + *Term Frequency*: How frequently a word appears in a document.
  + *Inverse Document Frequency*: How common or rare that word is across all documents.
* This step allows the model to represent documents numerically, emphasizing distinguishing words and down weighting common ones.

**4. Similarity Calculation**

* Using the processed TF-IDF matrix, the tool computes pairwise cosine similarity:

cosine similarity(A,B)=A⋅B∥A∥∥B∥\text{cosine similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}cosine similarity(A,B)=∥A∥∥B∥A⋅B

* A high value (close to 1) means documents share many important terms, while low scores (close to 0) indicate little overlap.

**5. Result Presentation and Visualization**

* The script outputs:
  + The top three most similar document pairs and their similarity scores.
  + A complete similarity matrix (heatmap) visualizing all pairwise scores. This allows users to identify clusters, outliers, or duplicates at a glance using matplotlib/seaborn.

**6. Automation and Extensibility**

* The use of modular scripts and Jupyter notebooks allows further analysis, tuning, or integration (for example, extending to new similarity metrics or larger datasets).

**7. Testing and Reliability**

* A test suite (test\_similarity.py) ensures the correctness of computations and supports reliable future development.

**Usage Steps**

1. Clone the repository and install dependencies using pip.
2. Run python src/main.py to execute the full workflow: preprocessing, similarity calculation, and visualization.

**Key Insights and Applications**

* The project demonstrates best practices in NLP for document comparison. TF-IDF and cosine similarity are gold-standard tools for textual analysis and clustering.
* The system supports large datasets via efficient matrix representations and can be adapted to business, academic, or research contexts needing document similarity assessment.