Abstract: Hybrid Text Retrieval Framework Using SBERT and Skip-Gram

This project introduces a system for extracting, processing, and analyzing text from PDF files to provide relevant answers to user questions. Built in Python and run on Jupyter Notebook, the system combines SBERT (Sentence-BERT) embeddings for sentence-level similarity and Skip-Gram modeling for word-level relevance. This hybrid approach efficiently handles large documents and ranks sentences by relevance to the user's query.

When a user enters the query the system calculates: Sentence-Level Similarity: Finds similar sentences in the document using SBERT. Word-Level Refinement: Fine-tunes the results using Skip-Gram word similarity. Sentences are ranked by combining the two scores, with an adjustable weight for fine-tuning.

Modules:

1. **PDF Text Extraction Module – extract_text_from_pdf(pdf_path)**
   Extracts text from PDF documents using PyMuPDF (fitz). This module processes each page and compiles the extracted text into a single document.
2. **Text Preprocessing Module – preprocess_text(text)**
   Cleans the extracted text by handling abbreviations, removing unwanted characters (e.g., numeric references), and structuring it for further analysis.
3. **Sentence Splitting Module – split_into_sentences(text)**
   Splits the preprocessed text into individual sentences for sentence-level processing and retrieval.
4. **SBERT Sentence Embedding Module – sbert_model = SentenceTransformer('all-MiniLM-L6-v2')**
   Converts sentences into dense vector representations to compute semantic similarity between the query and document sentences.
5. **Skip-Gram Model Training Module – train_skipgram(corpus)**
   Trains a Skip-Gram model on the extracted text to create word-level embeddings for understanding contextual relevance.
6. **Skip-Gram Similarity Computation Module – skipgram_similarity(query, sentence, tokenizer, word_embeddings)**
   Computes word-level similarity between the query and document sentences using cosine similarity on Skip-Gram embeddings.
7. **SBERT + Skip-Gram Combined Similarity Module – combined_similarity(query, corpus, corpus_embeddings, sbert_model, tokenizer, word_embeddings, alpha=0.7, top_k=5)**
   Computes the final similarity score using a weighted combination of SBERT (sentence-level) and Skip-Gram (word-level) similarity to retrieve the most relevant sentences.
8. **Execution & Testing Module**
   Runs the entire pipeline—extracting text, preprocessing, embedding, training, computing similarity, and retrieving the top relevant sentences based on the user query.

Libraries and Tools Used:

- PyMuPDF (fitz): Extracts text from PDF files.
- Regular Expressions (re): Cleans and preprocesses text.
- NumPy: Handles numerical operations like similarity calculations.
- Scikit-Learn: Measures similarity between vectors.
- Sentence-BERT (sentence_transformers): Creates sentence-level embeddings.
- TensorFlow: Tokenizes text and generates Skip-Gram word pairs.

Language and Platform:

- Language: Python 3.x
- Platform: Jupyter Notebook