**University Collaboration Recommendation System**

**Project Overview**

The University Collaboration Recommendation System is a tool designed to enhance academic partnerships by intelligently pairing students and professors with aligning research interests and fields. By harnessing the power of Natural Language Processing (NLP) and Machine Learning (ML), the system scrutinizes detailed research profiles to propose the most beneficial academic collaborations.

**Key Features**

* Robust Text Processing: Advanced NLP techniques are implemented to process and standardize research interest data.
* Semantic Matching: The system uses embeddings to understand the nuanced meanings within text, increasing the precision of matches.
* Visualization: WordClouds offer a visual representation of prevalent research topics shared by students and professors.
* Scalable Matching Engine: The engine utilizes cosine similarity to conduct scalable and effective similarity measurements.
* Interactive Analysis: Tools are available for an interactive exploration of data distributions and similarity assessments.

**Technologies Used**

The system is built using Python as the core programming language with the support of various libraries:

* Flask: For creating the web API and serving the front-end interface.
* Pandas & NumPy: For data manipulation and numerical operations.
* NLTK & SpaCy: For performing NLP tasks.
* Scikit-Learn: For ML operations and calculating cosine similarities.
* WordCloud & Seaborn: For generating visualizations.
* HTML & JavaScript: For developing the front-end user interface.

**Methodology and Libraries**

The recommendation system employs the following methodologies and libraries:

* Data Preprocessing: Using NLTK, text data undergoes cleaning and normalization.
* NLP Techniques: SpaCy is utilized for embedding the research interest text, grasping contextual meanings.
* Vectorization: Research interests are numerically represented via TF-IDF vectorization.
* Similarity Scores: Scikit-Learn is employed to determine similarity scores between profiles.
* Web API: Developed using Flask, it handles real-time HTTP requests for recommendations.
* Front-End Development: Crafted with HTML and JavaScript to enable user interactions with the API.

**Project Timeline**

The project was executed over an intense one-week period with the following breakdown:

* Day 1: Project initiation, literature review, data acquisition, and recommendation algorithm planning.
* Day 2-3: Coding the recommendation engine with data preprocessing and NLP feature engineering.
* Day 4: Implementing the similarity matching logic and engine integration.
* Day 5: Developing the Flask API, routing, and data handling. Constructing the front-end interface, API connectivity, and preliminary testing.
* Day 6: Checking improvement techniques like using SBERT for generating embeddings. Final GitHub deployment and documentation complete.

**Sources and Credits**

The development of this project was inspired by a combination of academic literature on recommendation systems, documentation of the various libraries used, and community-driven resources such as GitHub repositories. Some links -

* https://esource.dbs.ie/server/api/core/bitstreams/ea22d96a-262c-42bf-9bf3-8fbb98e3d36a/content
* https://aclanthology.org/R19-2009.pdf
* https://github.com/AmoliR/nlp-for-book-recommendation/tree/76ca80daf9eb733274a3d92887bbfbec7b48704c