**University Collaboration Recommendation System**

* Anushka sharma

**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.

Complete breakdown –  
1. Environment Setup and Data Loading

* The code begins by setting up the environment in a Google Colab notebook, mounting the Google Drive to access files, and installing necessary Python packages (spacy, sentence-transformers).
* Necessary libraries are imported, including data manipulation libraries (pandas, numpy), natural language processing libraries (nltk, spacy), and visualization tools (matplotlib, seaborn, WordCloud).
* NLTK resources are downloaded for text processing.
* Data is loaded from an Excel file containing separate sheets for students and professors.

2. Data Preprocessing

* Both student and professor dataframes are cleaned by stripping whitespace and converting text to lowercase.
* Missing values are filled with 'unknown'.
* Research interests, which are initially strings, are split into lists for easier manipulation.

3. Textual Data Visualization

* Word clouds are generated to visually represent the most common research interests among students and professors.
* The distributions of university fields among students and professors are visualized using bar plots.

4. Text Processing and Feature Engineering

* Text data (research interests) is processed using SpaCy to tokenize, remove stop words, and lemmatize.
* The processed text is then converted into embeddings using a pre-trained SpaCy model. Embeddings provide a numerical representation of text that captures semantic meanings.
* One-hot encoding is applied to categorical data (University Field) to transform it into a format suitable for modeling.

5. Feature Combination and Scaling

* Research interest embeddings and one-hot encoded university fields are combined into a single feature set for each individual.
* These combined features are scaled using StandardScaler to normalize the data, ensuring that no single type of feature dominates the others when computing similarities.

6. Similarity Computation

* Cosine similarity, a measure of the angle between two vectors, is calculated between every student and every professor. This similarity score indicates how closely related their research interests and university fields are.

7. Recommendation Generation

* Based on the similarity scores, top recommendations are generated for each student and professor. This is achieved by ranking professors for each student (and vice versa) according to their similarity scores and selecting the top matches.
* Additionally, recommendations among students and among professors are also generated using the same methodology, facilitating intra-group collaborations.

8. Output and Evaluation

* Recommendations are formatted and displayed for selected students and professors.
* The code includes placeholder functions for saving these recommendations to JSON files, allowing for persistent storage and later retrieval.
* Although the actual evaluation functions (for precision and recall) are commented out, the methodology suggests a way to evaluate the effectiveness of the recommendations by comparing them against a "ground truth" of known relevant matches.

9. Data Export

* Finally, recommendations are saved to JSON files, providing a structured way to output the recommendation data for use in other applications or for further analysis.

**Project Timeline**

The project was executed over 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