NLP Project Documentation

Semantic Job Recommendation System

1. Introduction

This project aims to develop a job recommendation engine based on user textual profiles. The goal is to analyze input information (experience, interests, qualities, skills, etc.) using a semantic embedding model to suggest the most relevant jobs.

2. Data Preprocessing

Before generating embeddings, a cleaning pipeline is applied to all user inputs to ensure linguistic consistency and reduce noise. The text is first converted to lowercase. Punctuation marks and special characters are removed using regular expressions, while English stopwords are filtered out with the *nltk.corpus.stopwords* library. Finally, redundant spaces are collapsed to produce a clean text format. Each text field (experience, interests, qualities) is processed independently, then converted into a cleaned list of textual inputs for embedding generation.

This ensures all textual information is clean and semantically coherent before being vectorized by the model.

3. Model Used

The multi-qa-mpnet-base-dot-v1 is a special model from Sentence-BERT (SBERT). This model was selected for semantic embedding generation, as it performs well for semantic similarity and textual analysis tasks. Embeddings are generated for user experience, interests, qualities, and skills. Skill levels are encoded as repeated tokens proportional to the user's self-assessed rating.

Each component is represented as a vector and combined through a weighted aggregation that reflects its relative importance. Interests receive the highest weight (0.4) as they capture the user's motivation and job affinity. Skills are highly weighted because they serve as a major differentiating factor between job roles. Experience is assigned a moderate weight (0.2) as it provides contextual background, while qualities are given a low weight (0.05) due to their limited discriminative power.

A final normalized embedding represents the full user profile.

4. Competency Blocks Management

A significant challenge in this project involves determining the optimal level of granularity for competency blocks. When the blocks are too broad, job descriptions tend to overlap excessively, causing similarity scores to converge. Conversely, if the blocks are too specific, users must provide an unnecessary level of detail. A balanced approach has therefore been adopted, maintaining general yet sufficiently discriminative competency representations.

5. Normalization and Adjustments

After combining all components, the resulting embedding vector undergoes L2 normalization to project it onto the unit sphere. Normalization ensures that all embeddings have the same magnitude, allowing cosine similarity to measure only the directional similarity (semantic closeness) rather than vector length. This improves consistency and stability across user profiles, regardless of text length or verbosity.

6. Interface Design

The user interface follows a minimalist design to ensure simplicity and clarity. It features a white background with a concise title and tagline. The system displays the results as a podium showing the top three recommended jobs. Each result includes a hyperlink that redirects the user to a popular job platform pre-filtered for the corresponding position.

7. Future Perspectives – Segmentation-Based Recommendation System

7.1 Current State

The system collects raw user inputs via a Streamlit interface and stores them for later use. These records are then used to develop and test code in Jupyter notebooks. Once validated, the notebook code is integrated back into the main analyse.py application for production..

7.2 Future Vision – Intelligent Cache via Clustering

The next development objective is to improve performance and reduce computational costs by eliminating redundant semantic computations for similar user profiles. The system will achieve this through an intelligent caching mechanism based on embedding clustering.

Each stored record will include the raw user input, the user's input (text and skill levels) embedded, the recommendation output (top three job suggestions), and associated metadata such as a timestamp and a unique identifier.

7.3 Processing Pipeline

When a new user profile is received, the system cleans and preprocesses the input text, generates an embedding, and then searches for a matching cluster in the cache. If a suitable cluster is found, the cached result is returned instantly. Otherwise, a full semantic analysis is performed, and the system updates or creates a new cluster. The resulting embedding and response are stored to enrich future recommendations.

7.4 Expected Benefits

This clustering and caching mechanism is expected to significantly enhance performance. Recommendation quality will also improve through consistent suggestions for similar profiles, while computational costs will be reduced.

7.5 Deployment Strategy

The deployment process is organized in three stages. First, precomputed user embeddings are collected from analyse.py and stored in a CSV. Second, these stored embeddings are used as historical data to perform the initial clustering for segmentation. Third, in the production environment, the system either returns the top 3 job recommendations associated with existing clusters or performs real-time analysis to create a new cluster if necessary. Optionally, the system can be further enhanced with advanced optimization, such as A/B testing, user feedback integration, and periodic reclustering to maintain adaptability.

7.6 Risks and Limitations

Despite its advantages, this approach presents several risks. Clustering bias may occur over time, making periodic reclustering necessary. The evolving nature of the job market requires frequent updates to competency blocks. Additionally, a cold start problem may arise initially, as the cache can only function effectively once a sufficient volume of data has been collected.

7.7. Conclusion

This evolution turns the current system—from real-time semantic analysis—into a hybrid intelligent solution, combining adaptive vector caching through clustering, full semantic analysis for atypical profiles, and continuous learning via cluster enrichment. The result is a scalable, fast, and self-improving job recommendation engine that ensures ever-increasing relevance and performance.