Summary of Approach, Key Findings, and Recommendations

Tudor-Cosmin Oanea

Approach

My approach for developing the recommendation system is centered on the utilization of user and product embeddings generated via a pre-trained transformer model. This model was selected due to its compact size and high ranking in benchmarking tests for embedding models (comparison can be found here). The core of the system is a Qdrant vector database that facilitates similarity searches, enabling the identification of products that closely align with the hypothetical preferences of users based on their historical interactions with products.

The implementation can be divided into the following stages:

1. **Data Embedding**: User and product data were embedded using the selected transformer model. The process involved transforming rows of user and product data into embeddable strings that captured key characteristics like product category, price, and user demographics.
2. **Dimensionality Reduction and Visualization**: A t-SNE algorithm was applied to reduce the high-dimensional embeddings into two dimensions, allowing for visual inspection and validation of the clustering of data points. This step was crucial in manually tuning the stringification process to ensure meaningful clusters for both users and products.
3. **Vector Storage and Similarity Search**: The embeddings were stored in Qdrant collections, enabling efficient similarity searches. I chose Qdrant for its ability to perform fast approximate nearest neighbor (ANN) searches locally, using the reliable HNSW algorithm.
4. **Recommendation Generation**: The recommendation logic involves constructing a hypothetical product embedding for a user based on the weighted average of products that similar users interacted with. This weighted approach considers both the nature of the interaction (e.g., view or purchase) and user ratings. The system then performs a similarity search on the product collection to identify potential recommendations, which are refined by filtering out previously purchased products and balancing the recommendation list between familiar and new products.

Key Findings

- **Model and Embedding Selection**: The choice of "dunzhang/stella_en_1.5B_v5" proved to be cost efficient and effective, allowing for high-quality embeddings that facilitated meaningful clustering of users and products, which is essential for accurate recommendations.
- **Stringification Process**: Through iterative testing and refinement, I empirically evaluated various stringification methods to determine the most effective way to encode user and product features into embeddable strings. The final method chosen demonstrated performance in clustering and similarity searches, improving the overall system's accuracy.

- **Clustering and Visualization**: Extensive experimentation with different parameter settings and visualization with t-SNE, validated that the chosen stringification and embedding strategies effectively captured the underlying relationships in the data. The resulting clusters showed products grouping by category (mainly) and brand and users grouping by age (mainly) and gender, confirming that the approaches successfully modeled the structure of the input.
- **Weighted Recommendation Strategy**: The weighted averaging approach to constructing the hypothetical product embedding allowed for a nuanced recommendation system that balances past user preferences with the potential for discovering new products.

Recommendations

1. **Data Enhancement**: The current implementation relies on synthetic data, which is useful for this PoC, but lacks the variability and richness of real-world data. For production deployment, it is crucial to integrate real-world datasets to further refine and validate the model's performance. Real data would likely improve the accuracy of user and product embeddings, resulting in more personalized and effective recommendations. Additionally, incorporating diverse data sources, such as user interaction histories, demographic information, and product metadata, could significantly enhance the model's ability to capture nuanced user preferences.
2. **Advanced Parameter Tuning**: The current system uses mostly empirically chosen thresholds and parameters for weight adjustments. These parameters could be further optimized by leveraging user feedback and conducting real-world A/B testing. Moreover, the system could benefit from a separate deep learning model that dynamically learns and fine-tunes these parameters based on evolving user behavior and preferences, leading to more adaptive and responsive recommendations.
3. **Scalability and Performance Optimization**: While Qdrant serves as a solid foundation for similarity searches, further exploration into alternative vector databases could yield performance improvements, especially at scale. For instance, Milvus offers a broader range of Approximate Nearest Neighbor (ANN) algorithms, which could be advantageous in scenarios where different search strategies are required. Evaluating Milvus or other vector databases could provide insights into optimizing search speed, memory usage, and scalability, ensuring the recommendation system can handle large volumes of data and high user traffic efficiently.
4. **Enhanced User Experience and Personalization**: To further refine the user experience, additional filtering and personalization logic should be considered. This could involve integrating contextual user data, such as current trends, seasonal preferences, and real-time behavior, to provide more relevant recommendations. Moreover, implementing a mechanism to explain why certain products are recommended could build user trust and engagement. For example, highlighting the specific attributes of products that align with the user’s past behavior or preferences could make the recommendations more transparent and personalized.
5. **Temporal Filtering of Recommendations**: To ensure that recommendations are relevant and reflect current user interests, the system could incorporate a filtering mechanism based on the timestamps of interactions (e.g., views, purchases, ratings) made by similar users and the user itself. By excluding products that the list of users

interacted with beyond a certain time threshold (e.g., more than a year ago), the system can avoid recommending outdated products that may no longer align with current trends or the user's evolving preferences. This temporal filtering enhances the relevance and timeliness of recommendations.

6. **Ethical Considerations and Bias Mitigation**: As the system evolves, it is important to address potential biases in recommendations, especially when using real-world data. Implementing fairness-aware algorithms and regularly auditing the recommendation outputs can help mitigate biases related to gender, age or other sensitive attributes. Ensuring ethical considerations are embedded in the development and deployment process will lead to a more inclusive and equitable recommendation system. This idea is related to the new EU AI ACT regulation.