Rental Recommendation System

CMPE-255 Project Report

Brandon Jacklyn
FNU Sameer
Sarvesh Borkar
Shruti Goyal

Problem Statement

Key Challenge:

- Finding rental properties that match specific preferences is time-consuming and overwhelming.
- Existing platforms lack support for nuanced attributes (e.g., proximity to places, quiet neighborhoods).

Limitations in Existing Solutions:

- Reliance on manual filters (dropdowns, checkboxes).
- Reliance on map-based search views.

Our Solution

Overview:

- A rental recommendation web service that enables users to search properties, and receive similar property recommendations.

Unique Features:

- Recommendations based on clustering.
- Automatic model re-deployment to live running web service.

System Architecture

Core Components:

- **Frontend**: React and Redux for modular, responsive UI.
- **Backend**: Python FastAPI for REST APIs and WebSockets, with MongoDB, PostgreSQL, and Redis.
- **Web Scraper Pipeline**: Apache Flink and Kafka for nightly scraping and data processing.
- **Recommendation Model**: Offline training in Google Colab, deployed to AWS S3.

Data Pipeline

Sources: Realtor.com (web scraping)

Preprocessing:

- Handling missing data using MICE Imputer.
- Feature engineering (creating new features e.g., total baths, has_pool).
- Dimensionality reduction with PCA, UMAP and Autoencoders.

Final Dataset:

- Rows reduced from 22,418 to 19,114 after preprocessing.

Recommendation Model Training

Key Steps:

1. Data loading and EDA (scatterplots, heatmaps, etc.).
2. Clustering (K-Means, Hierarchical, MeanShift, DBSCAN, HDBSCAN).
3. Evaluation using Silhouette score, Davies-Bouldin index, and Calinski Harabasz score.

Outcome:

- DBSCAN initially appeared to provide the best results, however, marked 90% of points as noise.
- HDBSCAN provided the actual best results with much few only ~10% noise points.

Deployment

Model Deployment Pipeline:

- Model and dimensionality reducer (UMAP/Autoencoder) are pickled to disk.
- Trained models uploaded to AWS S3 bucket in Colab.
- Backend periodically checks for and downloads latest model.
- Backend recomputes clusters from database updates cached recommendations.

Experiments and Results

Comparison: UMAP outperformed PCA and autoencoder for dimensionality reduction.
Results: Clusters were well separated by inspecting each cluster vs. its property type (like CONDO vs. SINGLE_FAMILY vs. APARTMENT, etc).
Visualizations: We have examples of cluster separation and property groupings.

Conclusion

Key Achievements:

- Built an end-to-end rental recommendation system.
- Improved the property search experience with a clustering model.
- Established a continuous data pipeline using Apache Flink, Kafka, and AWS S3.

Insights:

- Clustering algorithms struggle with too much data, dropping irrelevant columns and dimensionality reduction helped recommendations.
- Labeled data would greatly improve reasoning about model correctness.

Future Work

Enhancements:

- Real-time recommendation updates.
- Advanced chatbot NLP capabilities.
- Mobile application integration.
- Expanding data sources for richer diversity.

Thank You