Akshat Sharma

Midterm Project: Predicting Star Ratings on Amazon Movie Reviews

This project aims to predict the star ratings of Amazon movie reviews using traditional machine learning methods without deep learning or neural networks. I selected the XGBoost classifier for its robustness and compatibility with both numerical and text-based features. To facilitate processing, I used Google Colab and Google Drive due to the large dataset size, while parallelizing computations with the tqdm library to expedite preprocessing.

Data Exploration and Preprocessing

The dataset consists of fields like Product ID, User ID, Helpfulness Scores, Text, Summary, and Star Rating. Data quality was essential for model accuracy, so I performed extensive preprocessing:

- **Text Cleaning**: For Summary and Text fields, I removed HTML tags, converted text to lowercase, and applied stemming with NLTK's PorterStemmer. Stopwords were also removed to reduce noise, allowing the model to focus on meaningful words.
- **Feature Engineering**: New features were created to capture additional information:
- **Helpfulness Ratio**: Calculated as HelpfulnessNumerator / (HelpfulnessDenominator + 1), this feature represents perceived helpfulness.
- **Text Length**: Total number of characters in the Text field to capture review verbosity.
- **Caps Ratio**: The proportion of uppercase letters in Text, which could correlate with sentiment intensity.
- **Exclamation Count**: Number of exclamation marks in Text, representing emphatic sentiment.

This set of engineered features provided quantitative insights that complemented the text features and aimed to improve the model's performance.

Since some features were missing in the test data, I merged relevant columns from the training set to ensure consistency across both datasets. This preprocessing step minimized potential biases and overfitting risks, improving model generalizability on unseen data.

Model Preparation

To handle the large text fields, I applied **TF-IDF vectorization** to the Text field:

- **TF-IDF**: Used unigrams and bigrams with a cap of 3000 features to capture individual and paired words, preserving the most informative terms.
- **Feature Combination**: Combined the TF-IDF sparse matrix with the numeric features (helpfulness_ratio, text_length, caps_ratio, and exclamation_count) using scipy.sparse.hstack, creating a final feature set for model training.

Model Selection and Training

I chose XGBoost for this project due to its effectiveness with large, sparse feature sets and numerical data. The model's parameters were configured as follows:

- **Number of Estimators**: Set to 100 trees to balance performance with runtime.
- **Max Depth**: Restricted to 6 to prevent overfitting.
- **Learning Rate**: Set to 0.1 for smoother convergence.
- **n_jobs**: Set to -1 to utilize all available CPU cores and expedite training on Colab.

The training data was split into 80% training and 20% validation to simulate unseen data. This split enabled early identification of potential overfitting and provided insights into the model's robustness.

Evaluation and Results

The model's accuracy on the validation set was approximately 0.53. **Classification Report** metrics showed that certain ratings were occasionally misclassified as neighboring ratings (e.g., 4-star vs. 5-star ratings), which is reasonable given that subtle nuances in text can affect these closely related classes.

A **confusion matrix** revealed that higher ratings were often confused with adjacent classes, indicating areas for future improvement. Class imbalance may have contributed to this issue, as some ratings were overrepresented in the training data. Future iterations could apply SMOTE to address class imbalance.

Reflections and Future Directions

The helpfulness_ratio feature proved valuable, as it provided a measure of perceived review quality that correlated with rating accuracy. Experimenting with different values for model parameters (e.g., max_depth and n_estimators) provided insights into the balance between underfitting and overfitting.

In the future, I plan to explore additional models, like Random Forest, to compare their performance. This project underscored the importance of careful preprocessing, feature engineering, and model tuning when working with real-world data. Moving forward, I am also interested in testing more advanced algorithms like gradient-boosting models on large, structured datasets.

Citations

- Pandarallel. PyPI. (n.d.). Retrieved from https://pypi.org/project/pandarallel/
- ChatGPT. (n.d.). Retrieved from https://chatgpt.com/