Sentiment Analysis of Book Reviews

Nagaraja S B, Aviroop Karmakar, N Hariharan
MTech AI (Online)
IISc, Bangalore
Emails: nagarajasb@iisc.ac.in, aviroopk@@iisc.ac.in, nhariharan@@iisc.ac.in

***Abstract*—In the digital era, online reviews have become a cornerstone of customer decision-making, particularly in the book industry. Automating the analysis of these reviews is essential not only for summarizing customer opinions but also for offering strategic insights to authors and publishers. This report leverages machine learning techniques to classify book review sentiments and provides actionable insights.**

I. Introduction

In the digital era, online reviews have become a cornerstone of customer decision-making, particularly in the book industry. Platforms like Amazon host millions of reviews, providing valuable insights into a book's quality and reception. However, the sheer volume of reviews presents a challenge for potential readers, making it nearly impossible to extract a cohesive sentiment about a book efficiently. Automating the analysis of these reviews is essential not only for summarizing customer opinions but also for offering strategic insights to authors and publishers regarding a book's market potential. Leveraging proper machine learning techniques on book reviews can empower stakeholders with actionable insights aiding in enhancing customer experience and driving market understanding.

II. About the Dataset

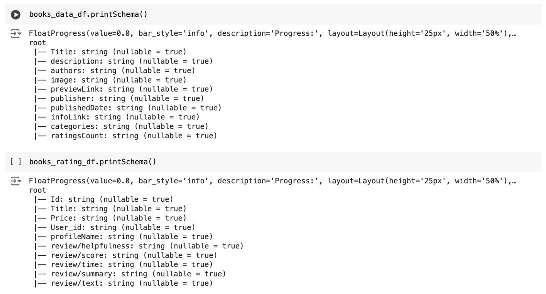
The dataset consists of two files:

- **Books_rating.csv:** Contains information about 3M book reviews for 212,404 unique books, including columns like Id, Title, Price, Review/Score, and more.
- **Books_data.csv:** Provides details about books such as genres, authors, descriptions, etc.

When a user posts a review on Amazon, they can provide both an elaborate review text and a concise summary. The size of the dataset is 2.7GB.

TABLE I: Dataset Summary

Dataset	Size
Books_rating.csv	2.7GB
Books_data.csv	172MB



```
books_data_df.printSchema()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- Title: string (nullable = true)
 |-- description: string (nullable = true)
 |-- authors: string (nullable = true)
 |-- image: string (nullable = true)
 |-- previewLink: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- publishedDate: string (nullable = true)
 |-- infoLink: string (nullable = true)
 |-- categories: string (nullable = true)
 |-- ratingsCount: string (nullable = true)
```

```
[ ] books_rating_df.printSchema()

FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- Id: string (nullable = true)
 |-- Title: string (nullable = true)
 |-- Price: string (nullable = true)
 |-- User_id: string (nullable = true)
 |-- profileName: string (nullable = true)
 |-- review/helpfulness: string (nullable = true)
 |-- review/score: string (nullable = true)
 |-- review/time: string (nullable = true)
 |-- review/summary: string (nullable = true)
 |-- review/text: string (nullable = true)
```

Fig. 1: Schema of the Dataset

III. Exploratory Data Analysis (EDA)

We leveraged Spark DataFrame for efficient handling and processing of the dataset, which was approximately 3GB in size. Spark's distributed computing capabilities enabled seamless analysis. The cluster configuration is as follows:

TABLE II: Cluster Configuration

Component	Cores	Memory (MB)
Hadoop	6	4000
Driver	2	2000
Executor	2	1000

Comprehensive EDA included:

- Examining data types, ranges, and frequency distributions.
- Visualizing trends with histograms, bar plots, and box plots.
- Identifying outliers and preprocessing areas for improvement.

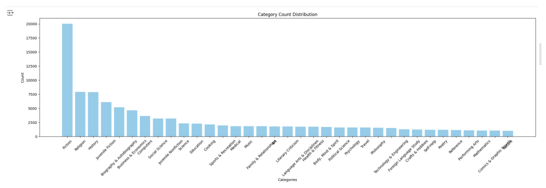
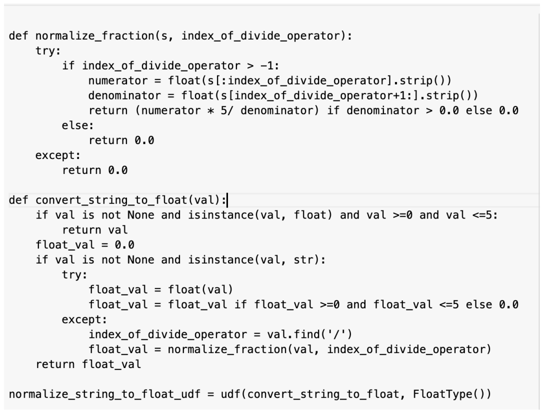


Fig. 2: Category Count Distribution

IV. Normalization and Feature Scaling

To standardize inconsistencies in `review/score` and `review/helpfulness`, a custom UDF was applied to normalize values into a consistent 0–5 scale. This ensured clean and reliable target labels for machine learning models.



```
def normalize_fraction(s, index_of_divide_operator):
    try:
        if index_of_divide_operator > -1:
            numerator = float(s[:index_of_divide_operator].strip())
            denominator = float(s[index_of_divide_operator+1:].strip())
            return (numerator * 5/ denominator) if denominator > 0.0 else 0.0
        else:
            return 0.0
    except:
        return 0.0

def convert_string_to_float(val):
    if val is not None and isinstance(val, float) and val >=0 and val <=5:
        return val
    float_val = 0.0
    if val is not None and isinstance(val, str):
        try:
            float_val = float(val)
            float_val = float_val if float_val >=0 and float_val <=5 else 0.0
        except:
            index_of_divide_operator = val.find('/')
            float_val = normalize_fraction(val, index_of_divide_operator)
    return float_val

normalize_string_to_float_udf = udf(convert_string_to_float, FloatType())
```

Fig. 3: Normalization Process

V. Data Sampling

Stratified sampling was employed to extract 5,000 data points, ensuring balanced representation of sentiment categories (positive, negative, neutral). Proportional representation was verified with summary statistics and visualizations.

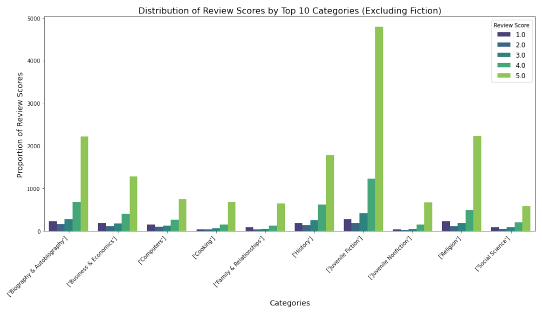


Fig. 4: Sampling Strategy

VI. Model Selection

A. Lexical Models

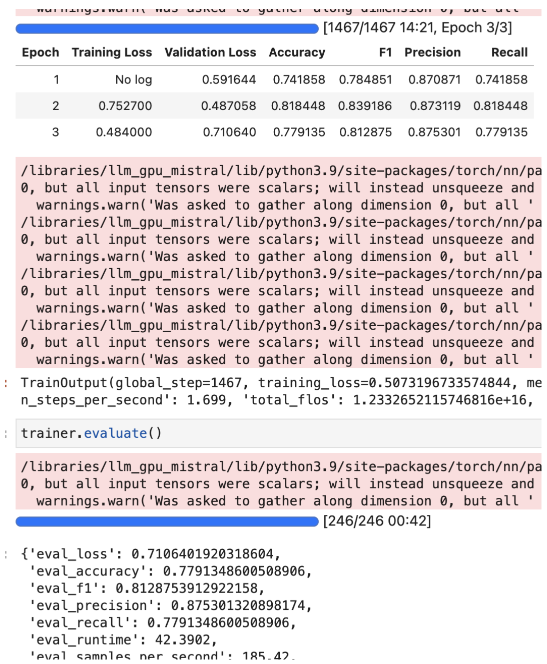
- **TF-IDF Vectorizer with Random Forest:** A baseline model for sentiment classification, leveraging word importance without considering word order or context.

B. Contextual Models

- **LSTM:** Processes sequential data, capturing dependencies and long-term context.
- **BERT:** A state-of-the-art transformer-based model capturing bidirectional context.

TABLE III: Model Performance Comparison

Model	Accuracy	Precision	F1 Score
TF-IDF + RF	72%	74%	73%
LSTM	75%	78%	75%
BERT	77%	87%	81%



```
  warnings.warn('Was asked to gather along dimension 0, but all
[1467/1467 14:21, Epoch 3/3]
Epoch  Training Loss  Validation Loss  Accuracy        F1  Precision    Recall
    1        No log         0.591644  0.741858  0.784851   0.870871  0.741858
    2      0.752700         0.487058  0.818448  0.839186   0.873119  0.818448
    3      0.484000         0.710640  0.779135  0.812875   0.875301  0.779135

/libraries/llm_gpu_mistral/lib/python3.9/site-packages/torch/nn/pa
0, but all input tensors were scalars; will instead unsqueeze and
  warnings.warn('Was asked to gather along dimension 0, but all '
/libraries/llm_gpu_mistral/lib/python3.9/site-packages/torch/nn/pa
0, but all input tensors were scalars; will instead unsqueeze and
  warnings.warn('Was asked to gather along dimension 0, but all '
/libraries/llm_gpu_mistral/lib/python3.9/site-packages/torch/nn/pa
0, but all input tensors were scalars; will instead unsqueeze and
  warnings.warn('Was asked to gather along dimension 0, but all '
/libraries/llm_gpu_mistral/lib/python3.9/site-packages/torch/nn/pa
0, but all input tensors were scalars; will instead unsqueeze and
  warnings.warn('Was asked to gather along dimension 0, but all '
: TrainOutput(global_step=1467, training_loss=0.5073196733574844, me
n_steps_per_second': 1.699, 'total_flos': 1.2332652115746816e+16,

: trainer.evaluate()

/libraries/llm_gpu_mistral/lib/python3.9/site-packages/torch/nn/pa
0, but all input tensors were scalars; will instead unsqueeze and
  warnings.warn('Was asked to gather along dimension 0, but all '
[246/246 00:42]

: {'eval_loss': 0.7106401920318604,
 'eval_accuracy': 0.7791348600508906,
 'eval_f1': 0.8128753912922158,
 'eval_precision': 0.875301320898174,
 'eval_recall': 0.7791348600508906,
 'eval_runtime': 42.3902,
 'eval_samples_per_second': 185.42,
```

Fig. 5: Bert Training Screenshot

VII. Pipeline Automation

To ensure a seamless and automated workflow for our project, we integrated all key steps—Exploratory Data Analysis (EDA), data sampling, model training, and evaluation—into a robust data pipeline using Apache Airflow. This pipeline allowed us to orchestrate these tasks in a modular and reusable way, ensuring that each step was executed in the correct sequence with proper dependencies.

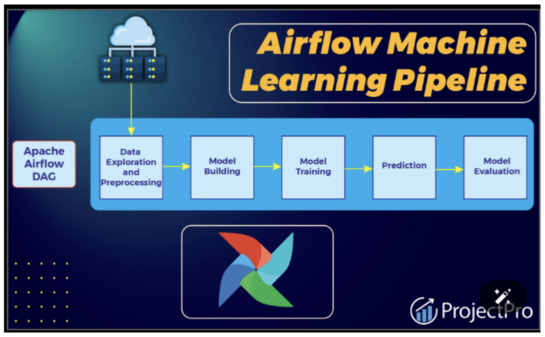


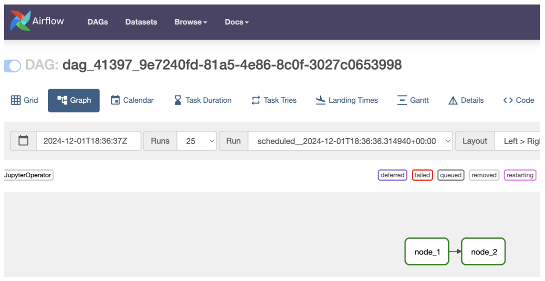
Fig. 6: Pipeline Architecture

A. Airflow DAG Instances

For this training pipeline, we set up two DAG instances:

TABLE IV: Airflow DAG Instances and Tasks

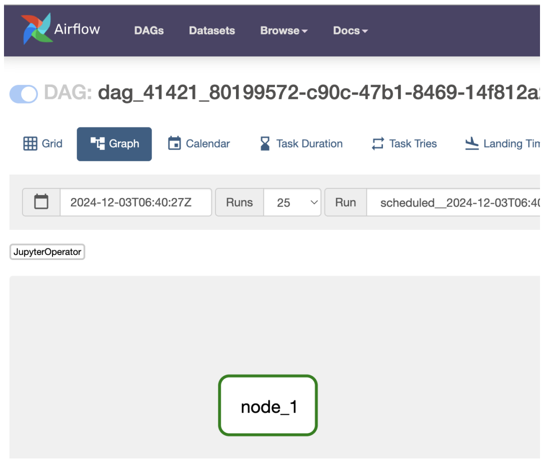
Instance	Tasks
Instance 1 *(num_of_retries = 2)*	• Jupyter Operator (Data Engineering / Preprocessing) • Jupyter Operator (Sampling)
Instance 2 *(num_of_retries = 2)*	• Jupyter Operator (Model Training and Evaluation)



DAG: dag_41397_9e7240fd-81a5-4e86-8c0f-3027c0653998

2024-12-01T18:36:37Z Runs 25 Run scheduled__2024-12-01T18:36:36.314940+00:00 Layout Left > Rig

Fig. 7: Screenshot: DAG Instance 1



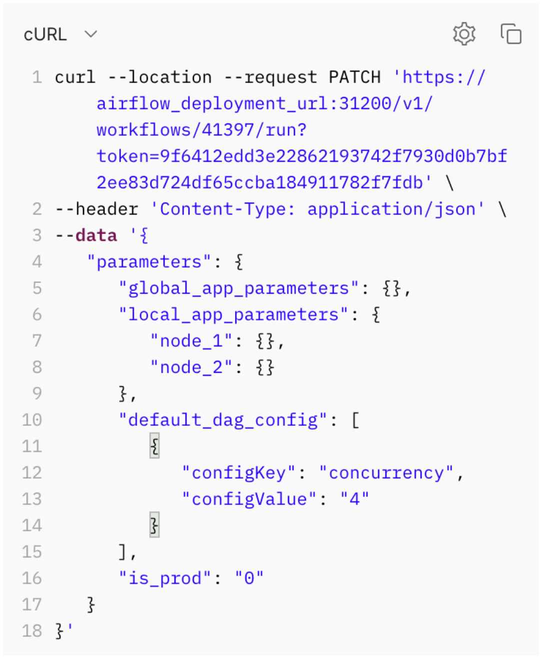
DAG: dag_41421_80199572-c90c-47b1-8469-14f812a

2024-12-03T06:40:27Z Runs 25 Run scheduled__2024-12-03T06:4

Fig. 8: Screenshot: DAG Instance 2

B. Pipeline Orchestration

The first DAG triggers the second DAG since they are hosted on different infrastructures (Instance 1 is used for CPU-based tasks, while Instance 2 is hosted on GPU-enabled infrastructure). The execution is triggered via a REST endpoint.



```
curl --location --request PATCH 'https://airflow_deployment_url:31200/v1/workflows/41397/run?token=9f6412edd3e22862193742f7930d0b7bf2ee83d724df65ccba184911782f7fdb' \
--header 'Content-Type: application/json' \
--data '{
   "parameters": {
      "global_app_parameters": {},
      "local_app_parameters": {
         "node_1": {},
         "node_2": {}
      },
      "default_dag_config": [
         {
            "configKey": "concurrency",
            "configValue": "4"
         }
      ],
      "is_prod": "0"
   }
}'
```

Fig. 9: Screenshot: REST Endpoint Trigger

C. Pipeline Execution Time

The total time taken for the entire pipeline to execute was approximately 20 minutes. This included data preprocessing, sampling, model training, and evaluation.

VIII. Challenges Faced

A. Challenges

- **Data Imbalance:** The dataset had a skewed distribution with more positive reviews, impacting model performance.
- **Overfitting:** The model struggled to generalize due to imbalanced data and overfitted on the training set.
- **Negation Handling:** Sentences with negations (e.g., ”not bad”) were misinterpreted, leading to incorrect predictions.
- **Ambiguity in Reviews:** Contradictory or ambiguous reviews (e.g., ”captivating storyline but dull writing”) were hard to classify.

B. Mitigation Strategies

- **Data Imbalance:** Used under-sampling and class-weight adjustments to balance the dataset.
- **Overfitting:** Applied regularization, cross-validation, and proper train-test splits.
- **Negation Handling:** Leveraged BERT to handle contextual subtleties.
- **Ambiguity in Reviews:** Applied data augmentation to improve model robustness.

IX. Conclusion

This report demonstrates the use of machine learning for sentiment analysis of book reviews. Future work will focus on fine-tuning models for text summarization and exploring additional data sources.