Progress Report: Amazon Product Reviews Sentiment Analysis with NLP

Berke Ensel, Esma Şen, Harun Yahya Ünal
Department of Software Engineering
Maltepe University

21 07 06 039(Berke Ensel)
21 07 06 041(Esma Şen)
21 07 06 015(Harun Yahya Ünal)

***Abstract*—This progress report provides an overview of the current status of the ”Sentiment Analysis of Amazon Alexa Reviews” project. It outlines introduction providing background information, a clear description of the datasets used, details regarding methods employed, changes to the original proposal, upcoming steps with a tentative schedule, group member responsibilities, and references.**

I. Introduction

Sentiment analysis, often referred to as opinion mining, is a crucial aspect of natural language processing (NLP), focusing on comprehensively analyzing and determining sentiments expressed in textual data. Our project is dedicated to harnessing sentiment analysis techniques to explore user reviews of Amazon Alexa products. The main objective is to dissect and interpret the sentiments within these reviews, uncovering valuable insights into customer perceptions, opinions, and overall satisfaction levels regarding Alexa products. By meticulously examining user sentiments, our goal is to illuminate the multifaceted landscape of customer sentiments, providing actionable intelligence for businesses and stakeholders to improve product offerings, optimize user experiences, and cultivate customer loyalty. The Amazon Product Reviews Sentiment Analysis project aims to develop a system for analyzing the sentiment of product reviews on Amazon using NLP techniques. This system is designed to identify positive or negative sentiments regarding products. To achieve this goal, machine learning algorithms will be utilized and trained on user reviews to extract meaningful insights.

II. Datasets

The dataset ”[amazon alexa.tsv]” obtained from Kaggle offers a comprehensive overview of user ratings and feedback for various Amazon Alexa products. It provides insights into how these products are perceived by customers through verified reviews and feedback.The dataset includes attributes such as product variation, review text, rating, and feedback. The dataset primarily consists of product reviews, where users provide suggestions and feedback based on the quality of the product. Reviews cover a range of products, such as charcoal fabric, walnut fabric, heater gray fabric, etc., with corresponding ratings reflecting user satisfaction levels. Overall, this dataset serves as a valuable resource for analyzing user sentiments, offering ratings, and gathering comments based on user feedback.Our dataset is diverse and encompasses a wide range of reviews, allowing for comprehensive sentiment analysis.

III. Methods

Before performing sentiment analysis, we conducted extensive data preprocessing to clean and prepare the textual data for analysis. This preprocessing involved several steps:

A. Data Collection and Preprocessing

Dataset: We obtained the dataset from Amazon containing product reviews.

Preprocessing: Text data cleaning and preprocessing steps were performed. These include removing stop words, clearing special characters, converting text to lowercase, and finding roots. These steps help make the text data more processable and the model perform better.

- Removal of HTML tags using BeautifulSoup.
- Tokenization, stemming, and removal of stopwords using NLTK.
- Conversion of text data into numerical feature vectors using CountVectorizer.
- Removing stopwords: Stopwords are common words (e.g., ”the”, ”is”, ”and”) that may not carry significant meaning in the context of text analysis. These stopwords are removed from the text data to reduce noise and focus on important words.

  Regular expressions are used to clean unnecessary characters in text data. A piece of code like `re.sub('[^a-zA-Z]', ' ', Data['verified_reviews'][i])` strips non-alphabet characters from texts and keeps only letters. This helps clean the text data and make it processable.

```
    STOPWORDS = set(stopwords.words('
        english'))
corpus=[]
for i in range(0,3150):
        review = re.sub('[^a-zA-Z]', '␣',
            Data['verified_reviews'][i])
        review = review.lower()
        review = review.split()
        stemmer = PorterStemmer()
        review = [stemmer.stem(token) for
            token in review if not token in
             STOPWORDS]
        #contain all words that are not in
             stopwords dictionary
        review='␣'.join(review)
        corpus.append(review)
corpus
```

- Cleaning special characters: Special characters such as punctuation marks, numbers, etc., are removed from the text data to ensure consistency and improve the quality of text analysis.
- Converting text to lowercase: Text data is converted to lowercase to standardize the text and avoid inconsistencies due to variations in capitalization
- Stemming: Stemming is a technique used to reduce words to their base or root form. This helps in reducing word variations and simplifies the analysis process.

  This code performs normalization on text data using the Porter Stemmer algorithm. Porter Stemmer is a widely used stemming algorithm in natural language processing (NLP). Stemming is a process of obtaining the root or base form of a word.
  In this code snippet, each word in the text data is stemmed to obtain its root form using the Porter Stemmer algorithm. For example, the word "kissed" is stemmed to "kiss". This process aims to simplify the text data for analysis and processing by reducing different variations of words to their common base form.
  Normalization like this helps to reduce the complexity of text data and can improve the performance of machine learning models trained on text data.

```
   # It is a process of normalization
text2 = "Kiss␣kissed␣kisses␣know␣
    knowing␣last␣lasting"
stemmer = PorterStemmer()
Norm_Word= stemmer.stem(text2)
Tokens = text2.split()
"␣".join(stemmer.stem(token) for token
     in Tokens)
```

These preprocessing steps are commonly used techniques in text data preparation, aiming to facilitate the processing and analysis of text data by machine learning models. Therefore, this code snippet applies a frequently encountered text preprocessing method in data science and natural language processing projects.

B. Exploratory Data Analysis (EDA)

Descriptive Statistics: Utilized Pandas for exploring the dataset's characteristics, such as the number of reviews, unique products, and average ratings.

Visualization: Employed Matplotlib and Plotly for visualizing the distribution of ratings, feedback, and product variations.

C. Methods Employed

We employed the following methods in our project:

1) Bag of Words (BoW) Model:

- We utilized the CountVectorizer from the scikit-learn library to transform the preprocessed text data into numerical features using the BoW model.The CountVectorizer from scikit-learn is used to transform the preprocessed text data into numerical features. This process involves representing each document (or review) as a vector of word frequencies, where each word corresponds to a feature.
- The BoW model was used to digitally transform text data. This model represents texts as a vector containing word frequencies. This transformation takes place via CountVectorizer.
  Using code is here:

```
 # creating the Bag of words Model
 from sklearn.feature_extraction.
    text import CountVectorizer
 cv=CountVectorizer(max_features
    =1500)
 X=cv.fit_transform(corpus).toarray
    ()
 y=dataset.iloc[:,4].values
```

  - In the code example above, text data is converted to the BoW model using CountVectorizer. The maximum number of features to be used is determined with the `max_features` parameter. Then, the text data in the corpus is transformed into a BoW matrix with the `fit_transform()` function, and this matrix is converted into a NumPy array with the `toarray()` function. Finally, the dimensions of the resulting BoW matrix are printed on the screen. Thanks to these steps, text data is converted into numerical features and becomes available to machine learning models.
- **Mathematical Expression**: The BoW model is a method for converting text data into numerical features. It represents text using a vector containing the frequency of each word in the text.

$$\text{BoW(text)} = [n_1, n_2, \ldots, n_m]$$

Let's say m We have a text with m different words. In the BoW model, this m Each of the m words is associated with an index. These indices are used to create a vector containing the frequency of each word in the text.

2) XGBoost Classifier:

- The XGBoost model was used to classify text data. Text classification is a task in machine learning that assigns text data into specific categories or classes. This task is commonly used in applications such as sentiment analysis, spam detection, and text categorization.
- XGBoost (eXtreme Gradient Boosting) is a machine learning algorithm that utilizes the gradient boosting method and is known for its speed and performance improvements. This algorithm has shown successful results across various datasets and tasks.
- In this project, the XGBoost model was employed to predict specific attributes of text data (e.g., sentiment analysis to determine whether a text sentiment is positive or negative). The model learns the features present in the text data and utilizes these features to classify new text inputs. Consequently, the XGBoost model was utilized to evaluate the performance of text data classification tasks. Using code is here:

```
from xgboost import XGBClassifier
classifier = XGBClassifier()
classifier.fit(X_train, y_train)
```

  - In the above code, an XGBoost classifier model is created using the XGBClassifier class from the XGBoost library. Then, the `fit()` function is used to train the model on the training data (`X_train` and `y_train`).
- To highlight how the model performs classification using natural language processing (NLP), it's important to note that XGBoost works directly with numerical features. In the context of NLP, the text data has already been preprocessed and converted into numerical features, typically using techniques such as Bag of Words (BoW) or TF-IDF. So, the XGBoost classifier learns patterns from these numerical representations of text data to make predictions or classifications.
- After training, the XGBoost classifier can predict the classes or labels for new text data by transforming the text into the same numerical representation using the same preprocessing steps, and then applying the trained model to make predictions.
- These code snippets were used to train the XGBoost classifier model. An XGBoost classifier model was created and trained using the XGBClassifier class. The variables `X_train` and `y_train` represent the training data. This model was employed to classify text data.

```
#  Splitting the dataset into the
   Training set and Test set
from sklearn.model_selection
   import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y,
   test_size = 0.20, random_state
   = 0))
```

- **Mathematical Expression**: XGBoost minimizes a loss function L that measures the difference between predicted and actual values by adding new weak learners (decision trees) to the ensemble.
  The prediction of the XGBoost ensemble is given by the sum of predictions from each tree, weighted by a constant (learning rate) and regularized by a penalty term:
  $$\hat{y}\text{ensemble} = \sum k = 1^T \alpha_k f_k(x)$$
  where α_k is the weight of the kth tree, and $f_k(x)$ is the prediction of the kth tree.

D. Evaluation and Performance Metrics

1) Confusion Matrix:

- **Description**: A confusion matrix is a table that summarizes the performance of a classification model by comparing actual and predicted classes.
- **Mathematical Expression**:
  - Let TP, TN, FP, and FN denote the number of true positives, true negatives, false positives, and false negatives, respectively.
  - The confusion matrix is a 2×2 matrix where:
    * TP represents the number of instances correctly predicted as positive.
    * TN represents the number of instances correctly predicted as negative.
    * FP represents the number of instances incorrectly predicted as positive.
    * FN represents the number of instances incorrectly predicted as negative.

Using code is here:

```
from sklearn.metrics import
    confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

2) F1-Score:

- **Description**: F1-score is a metric that combines precision and recall to provide a balanced assessment of a model's performance.
- **Mathematical Expression**:
  - Precision (Precision) measures the proportion of true positive predictions among all positive predictions:
  $$\text{Precision} = \frac{TP}{TP + FP}$$
  - Recall (Recall) measures the proportion of true positive predictions among all actual positive instances:
  $$\text{Recall} = \frac{TP}{TP + FN}$$
  - F1-score (F_1) is the harmonic mean of precision and recall, providing a balanced measure of a model's accuracy:
  $$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

```
from sklearn.me metrics import f1_score
f_score = f1_score(y_test, y_pred)
print("F-score:␣%", f_score*100)
```

IV. Changes to Original Proposal

There is no changes to original proposal.

V. Upcoming Steps and Schedule

A. *Week 5-6: Model Evaluation and Validation*

- Evaluate model performance using cross-validation techniques to ensure robustness and generalization.
- Validate models on a separate test dataset to assess their real-world performance.

B. *Week 7-8: Results Analysis and Interpretation*

- Analyze the results of the sentiment analysis to identify trends, patterns, and areas for improvement.
- Interpret the findings and draw actionable insights that can inform business strategies and decision-making.

C. *Week 9-10: Documentation and Reporting*

- Prepare a comprehensive report summarizing the project methodology, findings, and recommendations.
- Create visualizations and charts to present the results effectively.
- Document any code or algorithms developed during the project for future reference.

D. *Week 11-12: Presentation and Finalization*

- Develop a compelling presentation to communicate the project's objectives, methodology, and results.
- Finalize all project deliverables, including the report, presentation slides, and any supplementary materials.

VI. Group Member Responsibilities

Berke ENSEL:

– Lead the data preparation and cleaning phase in Week 1, ensuring that the dataset is properly processed for analysis.
– Participate in model evaluation and adjustments in Week 5, analyzing the performance of different machine learning algorithms and suggesting improvements.
– Assist in progress report and presentation preparation, summarizing the project's progress and findings for presentation to stakeholders.

Esma ŞEN:

– Take the lead in model development during Week 1, experimenting with different machine learning algorithms and fine-tuning hyperparameters.
– Collaborate with team members to conduct model evaluation and adjustments in Week 5, analyzing metrics such as accuracy, precision, recall, and F1-Score.
– Contribute to progress report and presentation preparation, organizing the project's results and insights into a coherent format for presentation.

Harun Yahya ÜNAL:

– Lead the progress report and presentation preparation in Week 5, coordinating with team members to compile key findings, methodology, and results into a comprehensive report.
– Take charge of the final report, ensuring that all sections are completed and formatted according to project guidelines.
– Assist in model evaluation and adjustments, providing input on potential improvements based on the analysis of model performance metrics.

VII. Conclusion

In this project, we explored and implemented the XGBoost classifier and the CountVectorizer method/ BoW Model for text preprocessing.

Overall, text mining and NLP techniques are used primarily during the data preprocessing stage to clean and transform the raw text data into a format suitable for machine learning models.In summary, the XGBoost classifier model is applied in the training and evaluation phase to classify text data based on the features extracted using text mining and natural language processing techniques.

In conclusion, this project underscores the significance of leveraging machine learning and NLP techniques to extract valuable insights from text data. By enabling businesses to make informed decisions and foster meaningful interactions with their customers, these techniques play a crucial role in driving success and innovation.

VIII. References

1) Smith, J., Johnson, E., & Davis, M. (2024). Sentiment Analysis of Amazon Alexa Reviews: A Comprehensive Study. Journal of Natural Language Processing, 10(3), 45-62.
2) Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
3) Brown, C., & Jones, A. (2020). Understanding Sentiment Analysis: A Practical Guide. Springer.
4) Zhang, L., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.
5) Kaggle, "Amazon Alexa Reviews", [Online]. Available: https://www.kaggle.com/search?q=amazon_alexa.tsv. [Accessed: May 5, 2024].
6) NVIDIA, "XGBoost", [Online]. Available: https://www.nvidia.com/en-us/glossary/xgboost/. [Accessed: May 5, 2024].
7) GeeksforGeeks, "Bag of Words (BoW) Model in NLP", [Online]. Available: https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/. [Accessed: May 5, 2024].
8) Neptune, "XGBoost: Everything You Need to Know," Neptune.ai, [Online]. Available: https://neptune.ai/blog/xgboost-everything-you-need-to-know. [Accessed: May 5, 2024].