Delving into News Categories with KDD

Shreyareddy Edulakanti

Knowledge Discovery in Databases (KDD) provides a comprehensive framework for the data mining process. This research paper explores the application of KDD to the NewsCategory-Dataset.v3 dataset, sourced from Kaggle, aiming to uncover patterns and categories within news articles.The subsequent interpretation and evaluation of discovered patterns can provide a deeper understanding of news narratives, the evolution of stories over time, and the influence of external factors on media coverage. By leveraging KDD on news datasets, researchers, journalists, and policymakers can gain a holistic view of the media landscape, ensuring more informed, objective, and data-driven decision-making in the realm of information dissemination and consumption.

1 Introduction

Knowledge Discovery in Databases (KDD) is an umbrella term that encompasses the entire data mining process. Utilizing the NewsCategoryDataset.v3 dataset from Kaggle, we aimed to identify patterns and categories within the contained news articles.

2 Methods

2.1 Selection

The KDD process starts with the selection of a dataset that aligns with the research objectives. Our dataset was loaded to extract relevant information.

```
 Code to load the dataset
news_df = pd.read_json("News_Category_Dataset_v3.json", lines=True)
```

2.2 Preprocessing

Description: This step ensures the data is cleaned and transformed, making it suitable for further analysis. It includes tasks like handling missing values, eliminating duplicates, and encoding categorical data.

Action: We inspected the dataset, handled potential missing values, and prepared the textual data for further processing.

```
 # Checking for missing values in the dataset
missing_values_news = news_category_df.isnull().sum()")

# Display missing values
missing_values_news
```

```
link                0
headline            0
category            0
short_description   0
authors             0
date                0
dtype: int64
```

There are no missing values in the NewsCategoryDataset.v3.json dataset, which is excellent.

Next, we'll proceed with some basic data cleaning. Since this is textual data, we'll: Ensure that the text columns (headline, $\text{short}_description, and authors)$ $are in deed of string type$.

Convert text data to lowercase for uniformity. Strip any leading or trailing whitespaces. Let's start with these preprocessing steps.

```
# Ensuring text columns are of string type and cleaning the textual data
text_columns = ['headline', 'short_description', 'authors']

for column in text_columns:
    news_category_df[column] = news_category_df[column].astype(str)
    news_category_df[column] = news_category_df[column].str.lower()
    news_category_df[column] = news_category_df[column].str.strip()

# Display the cleaned dataset's first few rows
news_category_df.head()
```

```
link  \
0  https://www.huffpost.com/entry/covid-boosters-...
1  https://www.huffpost.com/entry/american-airlin...
2  https://www.huffpost.com/entry/funniest-tweets...
3  https://www.huffpost.com/entry/funniest-parent...
4  https://www.huffpost.com/entry/amy-cooper-lose...

                                        headline   category  \
0  over 4 million americans roll up sleeves for o...  U.S. NEWS
1  american airlines flyer charged, banned for li...  U.S. NEWS
2  23 of the funniest tweets about cats and dogs ...     COMEDY
3  the funniest tweets from parents this week (se...  PARENTING
4  woman who called cops on black bird-watcher lo...  U.S. NEWS

                                  short_description               authors  \
0  health experts said it is too early to predict...  carla k. johnson, ap
1  he was subdued by passengers and crew when he ...        mary papenfuss
2  "until you have a dog you don't understand wha...         elyse wanshel
3  "accidentally put grown-up toothpaste on my to...      caroline bologna
4  amy cooper accused investment firm franklin te...         nina golgowski
```

```
        date
0 2022-09-23
1 2022-09-23
2 2022-09-23
3 2022-09-23
4 2022-09-22
```

The data has been successfully cleaned. The text columns, namely headline, short$_description, and authors, have been$

Now that preprocessing is complete, we'll move to the Transformation step of the KDD process.

2.3 Transformation

To tailor the dataset, we applied the TF-IDF (Term Frequency-Inverse Document Frequency) technique. Before vectorization, we visualized the distribution of news articles across various categories to understand the dataset's balance. Description: Data transformation tailors the dataset to enhance its quality or transform its structure, optimizing it for the modeling phase.

Action: For our textual data, this involved employing techniques such as tokenization, stemming, and TF-IDF vectorization.

Given that the primary feature of this dataset is textual data, we'll use techniques like TF-IDF (Term Frequency-Inverse Document Frequency) to transform the text into a format that can be ingested by machine learning algorithms.

But before that, it might be beneficial to understand the distribution of news articles across different categories. This will give us insights into whether the dataset is balanced or if there are categories with very few articles. Let's visualize the distribution of news categories in the dataset.

```
import matplotlib.pyplot as plt

# Distribution of news articles across different categories
category_counts = news_category_df['category'].value_counts()

# Plotting
plt.figure(figsize=(15,8))
category_counts.plot(kind='bar', color='skyblue')
plt.title('Distribution of News Articles Across Categories')
plt.ylabel('Number of Articles')
plt.xlabel('Category')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
The bar chart visualizes the distribution of news articles across different categories. We can

Having a balanced dataset can improve the performance of machine learning models, but addressi

As mentioned, we'll use the TF-IDF technique to transform the textual data (headline and short_

from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Combining 'headline' and 'short_description' into a single column for transformation
news_category_df['combined_text'] = news_category_df['headline'] + " " + news_category_df['sho

# Using TF-IDF to transform the combined textual data
tfidf_vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')  # Limiting to top
tfidf_matrix = tfidf_vectorizer.fit_transform(news_category_df['combined_text'])

# Displaying the shape of the resulting matrix
tfidf_matrix.shape
(209527, 5000)
```

The combined textual data (headline and short$_description)hasbeentransformedusingtheTF-IDFtechniqueintoamatrixwith$
209,527
rows (corresponding to the number of articles) and

5,000

columns/features. We limited the features to the top

5,000

for computational efficiency.

With the data transformed, we'll move on to the Data Mining step of the KDD process.

2.4 Data Mining

This stage involved applying machine learning algorithms to uncover patterns. Given our dataset's nature, our primary task was to predict a news article's category based on its content. We used a multinomial Logistic Regression model for this purpose. Description: At this juncture, specific algorithms and methods are employed to extract patterns, relationships, or knowledge from the processed data.

Action: We used clustering techniques to identify potential categories within the news headlines. In this step, we'll apply machine learning algorithms to extract patterns or knowledge from the data. Given the nature of our dataset, a common task would be to predict the category of a news article based on its content.

For the Data Mining step, we'll follow these sub-steps:

Feature Selection: As we've already limited our TF-IDF features to the top

5,000,

this step is essentially handled. Model Selection: We'll start with a simple model to set a baseline. Given the multi-class nature of our target variable (news category), a Logistic Regression model with a 'multinomial' setting might be a good starting point. Training: We'll split our data into training and testing sets and train our selected model on the training set. Validation: We'll validate the model's performance on the test set to understand its predictive power. Let's start by splitting our dataset into training and testing sets.

```
from sklearn.model_selection import train_test_split

# Splitting the data into training and testing sets (80% train, 20% test)
X = tfidf_matrix
y = news_category_df['category']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, strat

X_train.shape, X_test.shape
((167621, 5000), (41906, 5000))
The dataset has been successfully split into training and testing sets. We have 167,621 samples

Next, we'll proceed with training a multinomial Logistic Regression model on the training data

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Initializing and training the Logistic Regression model
logreg = LogisticRegression(solver='saga', multi_class='multinomial', random_state=42, max_iter
logreg.fit(X_train, y_train)

# Predicting on the test set
y_pred = logreg.predict(X_test)

# Calculating accuracy and displaying the classification report
accuracy = accuracy_score(y_test, y_pred)
class_report = classification_report(y_test, y_pred, target_names=y.unique())
accuracy, class_report

RESULT
(0.578890850952131,
 '               precision    recall  f1-score   support\n\n     U.S. NEWS       0.37      0.1

The multinomial Logistic Regression model achieved an accuracy of approximately 57.89%57.89% or

U.S. NEWS: Precision = 0.37, Recall = 0.16, F1-Score = 0.22
COMEDY: Precision = 0.35, Recall = 0.13, F1-Score = 0.19
PARENTING: Precision = 0.46, Recall = 0.32, F1-Score = 0.38 ...
```

(and so on for other categories)

While the accuracy might seem decent for a first attempt, there's definitely room for improvement. The precision, recall, and F1-scores vary considerably among the categories, indicating that our model performs better for some categories than for others.

2.5 Interpretation and Evaluation

After mining, the extracted patterns were evaluated. We assessed the model's performance across various news categories, visualized the confusion matrix, and analyzed the most influential words for some news categories.

Description: Post mining, the discovered patterns and knowledge are evaluated and interpreted to provide actionable insights.

Action: We assessed the clusters formed by our model and tried to interpret their significance in the context of news categories.

In the interpretation step, we aim to provide insights into the model's predictions, understand the key factors driving the decisions, and interpret the model's overall behavior.

Feature Importance: For our model (Logistic Regression), we can look at the coefficients to understand the importance of different words in predicting each category. Visualization of Results: We can visualize the confusion matrix to understand where the model is making the most mistakes. Analysis of Misclassifications: We can further drill down into the articles that were misclassified to understand potential reasons for the misclassification. This can provide insights into areas of improvement. Category-wise Insights: We can also look at the most important words or phrases for each category, giving us an understanding of what kind of words are more indicative of each news category. Let's start with visualizing the confusion matrix

```
import numpy as np

# Normalize the confusion matrix
conf_matrix_normalized = conf_matrix.astype('float') / conf_matrix.sum(axis=1)[:, np.newaxis]

# Plot confusion matrix
plt.figure(figsize=(15, 12))
sns.heatmap(conf_matrix_normalized, annot=True, cmap="YlGnBu", xticklabels=y.unique(), yticklab
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title('Normalized Confusion Matrix')
plt.show()
```

Here's the normalized confusion matrix for our model's predictions:

The diagonal elements represent the percentage of correct classifications for each category. The off-diagonal elements show where the model is mistaking one category for another. For example, you can observe that many articles in the "WELLNESS" category are correctly classified (darker shade on the diagonal), but there are also misclassifications where "WELLNESS" articles are classified as "STYLE BEAUTY" and vice-versa.

Such visualizations help in understanding which categories the model might be confusing with each other, suggesting areas where additional feature engineering or model tuning might be beneficial.

2.6 Consolidation

The final stage involved deriving actionable insights from the knowledge obtained. We recommended potential categories or themes for uncategorized news articles based on the clusters analyzed.

Description: The final stage involves deploying the knowledge or actionable insights into the operational systems, or in our case, making recommendations based on the discovered clusters.

Action: By analyzing the prominent terms in each cluster, we could recommend potential categories or themes for uncategorized news articles.

For logistic regression, the coefficients of the features (in our case, the terms or words) give an indication of the importance of that feature. A positive coefficient for a feature indicates that it increases the log-odds of the class, making it a strong predictor for that class.

Let's analyze the top 10 influential words for some of the news categories based on the model's coefficients:

```
# Function to get top features for a given class
def top_features_for_class(class_index, num_features=10):
    # Get the coefficients for the class
    coefficients = logreg.coef_[class_index]
    # Sort them in descending order and get the top features
    top_feature_indices = coefficients.argsort()[::-1][:num_features]
    top_features = [(feature_names[i], coefficients[i]) for i in top_feature_indices]
    return top_features
# Get top features for some categories
categories = y.unique()
top_features_dict = {}
for category in categories:
    class_index = np.where(logreg.classes_ == category)[0][0]
    top_features_dict[category] = top_features_for_class(class_index)
top_features_dict

RESULT
{'U.S. NEWS': [('said', 5.142412498703175),
  ('covid', 4.974019089347927),
  ('california', 4.507296124573982),
  ('coronavirus', 4.438486019639785),
  ('shooting', 3.9271653690803032),
  ('police', 3.825475984012672),
  ('border', 3.6722860688040906),
  ('biden', 3.529549828289786),
  ('killed', 3.3628568155609098),
  ('officials', 3.2602584105448686)],
 ]}
Here are the top 10 influential words for a selection of news categories:

U.S. NEWS:

said
covid
california
coronavirus
```

```
shooting
police
border
biden
killed
officials
COMEDY:

snl
maher
colbert
trump
conan
oliver
trevor
meyers
comedy
corden
PARENTING:

kids
parenting
children
breastfeeding
baby
parents
babies
daughter
parent
son
... and so on for other categories.
```

These words serve as strong indicators for their respective categories. For instance, words like "snl", "maher", and "colbert" are influential in classifying an article under the COMEDY category. Similarly, words like "kids","parenting", and "children" are indicative of the PARENTING category.

This information can be helpful in understanding the nature of the articles under each category and can also be used to further refine the model if needed. For instance, if certain influential words are not representative of the category,it might be worth revisiting the data preprocessing steps or considering additional features.

3 Results

3.1 Data Preprocessing

The dataset had no missing values. Textual data columns were cleaned, ensuring uniformity in data representation.

3.2 Data Transformation

A visual representation of the distribution of news articles across categories revealed some imbalance. The textual data was then transformed using the TF-IDF technique, resulting in a matrix representation suitable for the machine learning process.

3.3 Data Mining

We achieved an accuracy of approximately

```
57.89%
```

using the Logistic Regression model on our test dataset.

3.4 Interpretation and Evaluation

The confusion matrix helped identify categories that were often confused with one another. The model's coefficients provided insights into the most influential words for predicting various news categories.

4 Discussion

While the achieved accuracy serves as a decent starting point, further optimization is possible. The precision, recall, and F1-scores varied across categories, indicating room for model improvement. Future work could involve exploring different machine learning models, handling class imbalance, and incorporating additional feature engineering techniques.

5 Acknowledgements


```
The dataset used in this research, News_Category_Dataset_v3,
was sourced from Kaggle. We extend our gratitude to the platform and the dataset's creators.
```