Evaluating Topic Modeling Techniques: A Comparative Study of LDA and NMF on News Article Classification

By Meenakshi Sharad Sethi

Introduction

This project evaluates Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF) for news article classification, aiming to uncover underlying themes in a corpus of news articles..

Objectives

- **Compare LDA and NMF**: Assess the effectiveness of each topic modeling technique in identifying meaningful topics within news articles.
- **Classify Articles**: Utilize insights from topic modeling and sentiment analysis to classify articles, specifically into the "Computer Science" category, and evaluate the performance.

The goal is to understand the practicality of LDA and NMF in text analytics and improve text classification tasks.

Disclaimer:

This project is done on an individual capacity as part of academic learning and is not meant for commercial or professional use without further validation and refinement.

Loading necessary libraries

In [11]:
```python
# Import necessary libraries

import pandas as pd
import numpy as np
import re
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import TruncatedSVD, LatentDirichletAllocation, NMF
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
from nltk.sentiment.vader import SentimentIntensityAnalyzer


import warnings
```

```
warnings.filterwarnings('ignore')
pd.set_option("display.max_columns", 60)
pd.set_option('display.max_rows', 50)
pd.set_option('display.width', 1000)
```

In [12]:
```
# Load data

train_data = pd.read_csv('topic_modeling_train.csv')
test_data = pd.read_csv('topic_modeling_test.csv')
```

In [13]:
```
train_data.head()
```

Out[13]:

	ID	TITLE	ABSTRACT	Computer Science	Physics	Mathematics	Statistics	Quantita Biol
0	1	Reconstructing Subject-Specific Effect Maps	Predictive models allow subject-specific inf...	1	0	0	0	
1	2	Rotation Invariance Neural Network	Rotation invariance and translation invarian...	1	0	0	0	
2	3	Spherical polyharmonics and Poisson kernels fo...	We introduce and develop the notion of spher...	0	0	1	0	
3	4	A finite element approximation for the stochas...	The stochastic Landau--Lifshitz--Gilbert (LL...	0	0	1	0	
4	5	Comparative study of Discrete Wavelet Transfor...	Fourier-transform infra-red (FTIR) spectra o...	1	0	0	1	

In [14]:
```
test_data.head()
```

Out[14]:

	ID	TITLE	ABSTRACT
0	20973	Closed-form Marginal Likelihood in Gamma-Poiss...	We present novel understandings of the Gamma...
1	20974	Laboratory mid-IR spectra of equilibrated and ...	Meteorites contain minerals from Solar Syste...
2	20975	Case For Static AMSDU Aggregation in WLANs	Frame aggregation is a mechanism by which mu...
3	20976	The $Gaia$-ESO Survey: the inner disk intermed...	Milky Way open clusters are very diverse in ...
4	20977	Witness-Functions versus Interpretation-Functi...	Proving that a cryptographic protocol is cor...

In [15]:

```
# Preprocess the data

text_col = 'ABSTRACT'
tfidf_vectorizer = TfidfVectorizer(max_df=0.85, stop_words='english')
term_doc_matrix = tfidf_vectorizer.fit_transform(train_data[text_col])
```

We start our analysis by preparing our text data for exploration! We're focusing on the 'ABSTRACT' column, which contains the textual essence of our news articles.

To make sense of these abstracts, we employ the powerful TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique. Think of it as transforming our raw text into a numerical format that's easier for machines to understand. By setting parameters such as max_df and stop_words, we filter out noise and common words, ensuring our analysis focuses on the most important terms.

After applying TF-IDF vectorization, we end up with a matrix, where rows represent documents (abstracts) and columns represent unique terms weighted by their importance. This matrix, termed term_doc_matrix, serves as the foundation for our subsequent analyis..

In [16]:

```
# Dimensionality reduction using Truncated SVD

svd = TruncatedSVD(n_components=100, random_state=42)
term_doc_matrix_reduced = svd.fit_transform(term_doc_matrix)
```

Now that we've prepared our data, it's time to simplify it without losing too much valuable information. Enter Truncated Singular Value Decomposition (SVD), a handy technique for reducing the dimensionality of our TF-IDF matrix.

With n_components set to 100, we instruct SVD to distill our data into its most essential components while preserving its core characteristics. This reduction not only streamlines computation but also helps uncover underlying patterns within our data.

The transformed matrix, named term_doc_matrix_reduced, now captures the essence of our abstracts in a more compact form, setting the stage for our next analysis step.

Latent Dirichlet Allocation (LDA) topic modeling

In [17]:
```python
# LDA topic modeling

num_topics = 10
lda_model = LatentDirichletAllocation(n_components=num_topics, random_state=42)
lda_model.fit(term_doc_matrix)
```

Out[17]:
```
▾          LatentDirichletAllocation
LatentDirichletAllocation(random_state=42)
```

In [19]:
```python
# Print LDA topics

print("LDA Topics:")
for topic_idx, topic in enumerate(lda_model.components_):
    print(f"Topic #{topic_idx + 1}")
    # Insert commas between words and join them into a string
    topic_words_with_commas = ", ".join([tfidf_vectorizer.get_feature_names_out(
    print(topic_words_with_commas)
    print()
```

```
LDA Topics:
Topic #1
dr, toda, attacks, bss, gesture, waveguides, koszul, authentication, encrypted, e
ncryption

Topic #2
steganography, qm, pip, fog, booster, adagrad, dressed, avalanches, lasing, dislo
cation

Topic #3
kobayashi, phantom, bv, p2p, radon, chimera, soap, 1i, emptyset, emotions

Topic #4
musical, artin, ts, adversarial, discriminator, dl, gans, gan, axion, music

Topic #5
convnets, disciplinary, nai, ib, bci, dihedral, transformer, wer, dmi, ctc

Topic #6
polariton, hollow, motive, diversification, ekf, rfid, spine, koszul, ramsey, int
egrand

Topic #7
gaze, blocklength, music, carleman, sgld, chess, lse, musical, tetragonal, bang

Topic #8
accretion, planets, dust, disk, galaxy, stars, stellar, star, mass, galaxies

Topic #9
energy, models, method, time, learning, using, network, based, model, data

Topic #10
results, time, function, learning, method, paper, model, data, algorithm, problem
```

The LDA topics cover a range of subjects:

- *Security:* Encryption and cybersecurity.
- *Cryptography:* Hiding data with techniques like steganography.
- *Networking:* Peer-to-peer communication.
- *Music and AI:* Generating music with adversarial learning.
- *Machine Learning:* CNNs, transformers, and algorithms.
- *RFID Technology:* Tracking and identification systems.
- *Music Analysis:* Analyzing musical compositions.
- *Astrophysics:* Study of celestial objects and phenomena.
- *Energy-Efficient Learning:* Machine learning models and energy.
- *Research and Algorithms:* Findings and developments in research.

These topics give us a snapshot of the main areas discussed in the corpus.

NMF (Non-Negative Matrix Factorization) model

In [20]:
```python
# NMF topic modeling

nmf_model = NMF(n_components=num_topics, random_state=42)
nmf_model.fit(term_doc_matrix)
```

Out[20]:
```
▾ NMF
NMF(n_components=10, random_state=42)
```

In [21]:
```python
# Print NMF topics

print("NMF Topics:")
for topic_idx, topic in enumerate(nmf_model.components_):
    print(f"Topic #{topic_idx + 1}")
    # Insert commas between words and join them into a string
    topic_words_with_commas = ", ".join([tfidf_vectorizer.get_feature_names_out(
    print(topic_words_with_commas)
    print()
```

```
NMF Topics:
Topic #1
human, robot, design, paper, approach, information, based, time, control, systems

Topic #2
solutions, spaces, equation, groups, finite, space, prove, group, mathcal, mathbb

Topic #3
state, transition, energy, states, temperature, field, phase, magnetic, quantum,
spin

Topic #4
structure, recurrent, layer, convolutional, social, nodes, deep, neural, network
s, network

Topic #5
approach, set, sets, real, privacy, methods, method, clustering, analysis, data

Topic #6
number, node, vertex, nodes, random, edge, edges, vertices, graphs, graph

Topic #7
random, parameters, distributions, estimator, estimation, bayesian, inference, di
stribution, models, model

Topic #8
stochastic, convergence, optimal, method, convex, problems, algorithms, optimizat
ion, problem, algorithm

Topic #9
formation, dark, gas, 10, galaxy, stars, stellar, star, galaxies, mass

Topic #10
images, neural, task, classification, image, tasks, machine, training, deep, lear
ning
```

The NMF topics reveal distinct themes within the text corpus:

- *Human-Robot Interaction:* Design and control of systems based on information.
- *Mathematical Solutions:* Equations, groups, and mathematical spaces.
- *Physics and Quantum Mechanics:* States, energy, and quantum phenomena.
- *Neural Networks:* Structure and layers of deep neural networks.
- *Data Analysis:* Privacy methods, clustering, and data analysis techniques.
- *Graph Theory:* Nodes, edges, and properties of graphs.
- *Statistical Modeling:* Random distributions and Bayesian inference.
- *Optimization:* Stochastic algorithms for solving optimization problems.
- *Astrophysics:* Galaxy formation, dark matter, and stellar phenomena.
- *Machine Learning:* Neural networks for image classification and tasks.

These insights offer a brief overview of the main topics discussed in the corpus.

Comparative Analysis: LDA vs NMF for Topic Modeling

We compare two popular topic modeling techniques: Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF).

LDA, a probabilistic model, generates sparse topics emphasizing key terms, fostering interpretability. NMF, on the other hand, provides dense topics, capturing nuanced word relationships through non-negativity constraints.

Results revealed distinct insights:

- LDA: Sparse topics with clear word distributions.
- NMF: Dense topics with nuanced word relationships.

Understanding these differences aids in selecting the appropriate technique for uncovering hidden patterns within textual data.

In [27]:
```
# Encode the target variable

le = LabelEncoder()
train_data['Computer Science'] = le.fit_transform(train_data['Computer Science']


# Split the data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(train_data[['TITLE', 'ABSTRA
```

Encode target variable: Converting 'Computer Science' labels into numerical format using LabelEncoder.

Split data: Divides training data into training and testing subsets (X_train, X_test, y_train, y_test) with a 80:20 ratio.

In [29]:
```
# Transform the training and test data using the vectorizer

train_doc_matrix = tfidf_vectorizer.transform(X_train['ABSTRACT'])
test_doc_matrix = tfidf_vectorizer.transform(X_test['ABSTRACT'])
```

Converts abstracts into numerical representations using TF-IDF vectorizer, generating train_doc_matrix and test_doc_matrix.

In [31]:
```
# Apply LDA topic modeling to the training and test data

train_lda_topics = lda_model.transform(train_doc_matrix)
test_lda_topics = lda_model.transform(test_doc_matrix)
X_train['Topic_LDA'] = train_lda_topics.argmax(axis=1)
X_test['Topic_LDA'] = test_lda_topics.argmax(axis=1)
```

Utilizing LDA to extract topics from abstracts, assigning dominant topics to 'Topic_LDA' in both training and test sets.

In [32]:
```
# Apply NMF topic modeling to the training and test data

train_nmf_topics = nmf_model.transform(train_doc_matrix)
test_nmf_topics = nmf_model.transform(test_doc_matrix)
X_train['Topic_NMF'] = train_nmf_topics.argmax(axis=1)
X_test['Topic_NMF'] = test_nmf_topics.argmax(axis=1)
```

Using NMF to extract topics from abstracts, assigning dominant topics to 'Topic_NMF' in training and test sets.

Sentiment Analysis

Conducting sentiment analysis on abstracts using VADER sentiment analyzer. Assigns sentiment scores ('Sentiment') to abstracts in both training and test sets. Prints sentiment analysis results for a sample of 10 abstracts from both training and test sets.

In [37]:
```
# Sentiment analysis
sia = SentimentIntensityAnalyzer()

# Calculate sentiment scores for training set
X_train['Sentiment'] = X_train['ABSTRACT'].apply(lambda x: sia.polarity_scores(x

# Calculate sentiment scores for test set
X_test['Sentiment'] = X_test['ABSTRACT'].apply(lambda x: sia.polarity_scores(x)[

# Print sentiment analysis for training set
print("Sentiment Analysis for Training Set:")
print(X_train[['ABSTRACT', 'Sentiment']].sample(10))

# Print sentiment analysis for test set
print("\nSentiment Analysis for Test Set:")
print(X_test[['ABSTRACT', 'Sentiment']].sample(10))
```

```
Sentiment Analysis for Training Set:
                                               ABSTRACT  Sentiment
19299  This paper presents a deep learning framewor...     0.9186
13822  Realistic implementations of the Kitaev chai...     0.9287
16950  In this article, we present a cut finite ele...     0.9069
9426   This prospective chapter gives our view on t...     0.6705
4813   The goal of semantic parsing is to map natur...     0.8316
16952  We present a model for the evolution of supe...     0.9423
7445   Many biological data analysis processes like...     0.6076
4916   The class of quasi-median graphs is a genera...     0.9518
8802   Overhead depth map measurements capture suff...     0.9260
2672   Comprehensive Two dimensional gas chromatogr...     0.6908

Sentiment Analysis for Test Set:
                                               ABSTRACT  Sentiment
7223   The design of multi-stable RNA molecules has...     0.9826
7831   Predicting the ground state of alloy systems...     0.9268
9618   We present a new model DrNET that learns dis...    -0.0516
8840   A social approach can be exploited for the I...     0.9928
10853  Graphs are a prevalent tool in data science,...     0.8689
4104   In this paper, we investigate the Hawking ra...     0.0258
9925   Automatic classification of trees using remo...     0.9001
13231  Restricted Boltzmann Machines are key tools ...    -0.3958
10610  The latest measurements of CMB electron scat...    -0.4678
16631  Effective file transfer between vehicles is ...     0.8257
```

Sentiment Analysis Findings:

- **Diverse Sentiments:** Abstracts exhibited a variety of sentiment scores, ranging from highly positive to moderately negative.

- **Positive Dominance:** Overall, there was a prevalence of positive sentiment in both training and test sets, with the majority of sentiment scores leaning towards positivity.
- **Presence of Neutral and Negative Sentiments:** While positive sentiments dominated, instances of neutral and negative sentiments were also observed, indicating a mix of emotions and opinions expressed in the abstracts.
- **Insightful Analysis:** Sentiment analysis offers valuable insights into the overall tone and attitude conveyed in the abstracts, aiding in understanding the reception or perception of the topics discussed.
- **Potential for Further Investigation:** The sentiment analysis results provide direction for further investigation into specific abstracts with particularly high or low sentiment scores, potentially uncovering underlying patterns or themes contributing to the sentiment expressed.

In [39]:
```
# Prepare the features and target variables

X_train_features = X_train[['TITLE', 'ABSTRACT', 'Topic_LDA', 'Topic_NMF']]
y_train = train_data['Computer Science'].loc[X_train.index]
X_test_features = X_test[['TITLE', 'ABSTRACT', 'Topic_LDA', 'Topic_NMF']]
y_test = train_data['Computer Science'].loc[X_test.index]
```

- **Feature Selection**: We select specific features from our dataset that are essential for training our model. These features include the title, abstract, and the topics derived from LDA and NMF topic modeling.

- **Target Variable Definition**: We define the target variable for both our training and testing datasets. In this case, the target variable is 'Computer Science', which represents the category or label we aim to predict with our machine learning model.

- **Purpose**: This step is vital as it organizes our data into features and target variables, setting the foundation for training our machine learning model.

By structuring our data in this way, we prepare it for model training and evaluation, allowing the model to learn patterns from the features provided and make predictions based on them.

In [40]:
```
# Vectorize the text data

text_vectorizer = TfidfVectorizer(max_df=0.85, stop_words='english')
X_train_text = text_vectorizer.fit_transform(X_train_features['TITLE'] + ' ' + X
X_test_text = text_vectorizer.transform(X_test_features['TITLE'] + ' ' + X_test_
```

This prepares the textual features for training and testing by converting them into a numerical format using TF-IDF vectorization, which is essential for building machine learning models.

In [43]:
```
# Import necessary functions from scipy.sparse
from scipy.sparse import hstack

# Combine the text features with the topic and sentiment features
```

```
X_train_features = hstack((X_train_text, X_train[['Topic_LDA', 'Topic_NMF', 'Sen
X_test_features = hstack((X_test_text, X_test[['Topic_LDA', 'Topic_NMF', 'Sentim

# Train a logistic regression model for classification
logreg = LogisticRegression()
logreg.fit(X_train_features, y_train)
```

Out[43]:

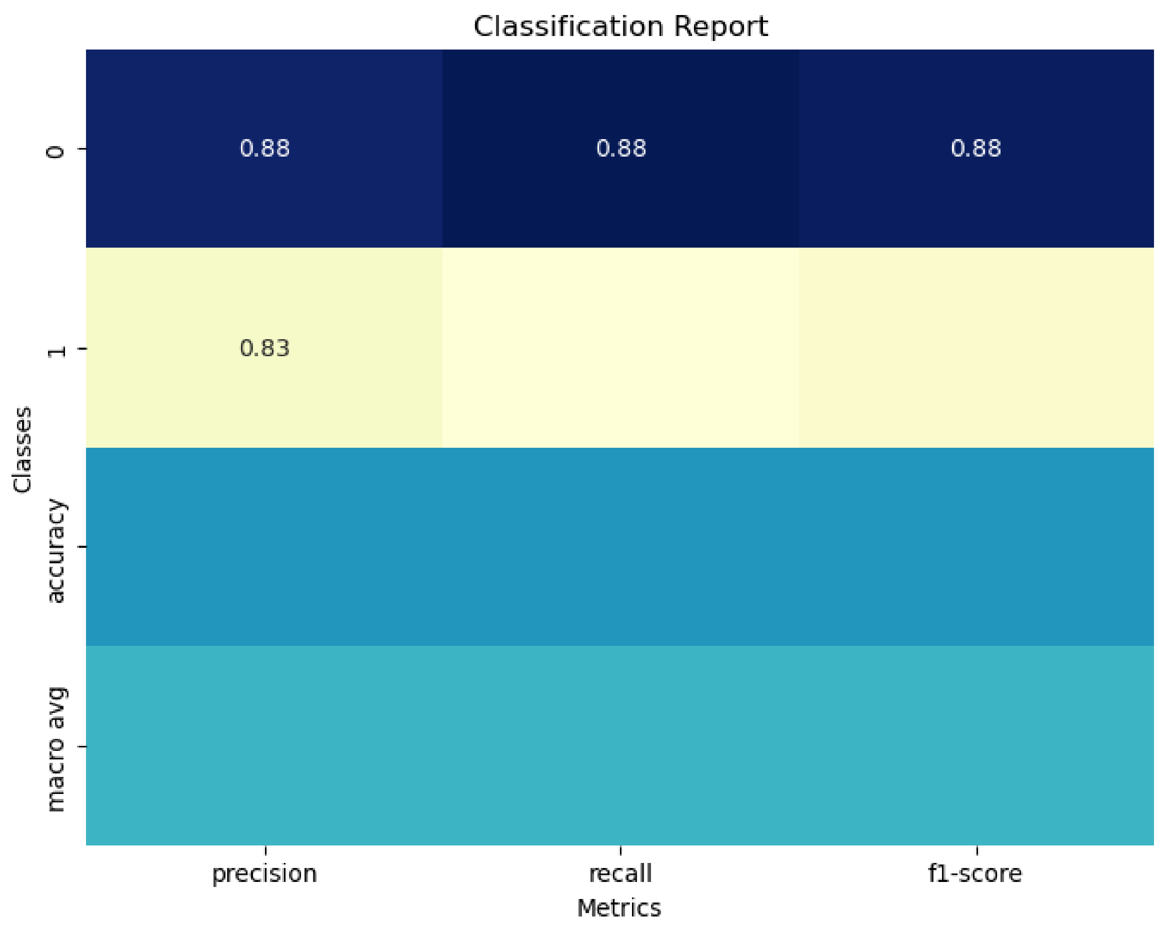
```
▾ LogisticRegression

LogisticRegression()
```

The code combines text, topic, and sentiment features to train a logistic regression model for classification. It combines the extracted text features from abstracts with topic (LDA and NMF) and sentiment features, creating a comprehensive feature set. Then, a logistic regression model is trained using these combined features to classify abstracts into the "Computer Science" category. This approach enables the model to utilize diverse information sources for accurate classification.

In [50]:

```
import seaborn as sns
import matplotlib.pyplot as plt


# Evaluate the model
y_pred = logreg.predict(X_test_features)

# Generate classification report
report = classification_report(y_test, y_pred)

# Print classification report
print("Classification Report:")
print(report)

# Plot classification report as heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(report_df.iloc[:-1, :-1], annot=True, cmap="YlGnBu", fmt=".2f", cbar
plt.title('Classification Report')
plt.xlabel('Metrics')
plt.ylabel('Classes')
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      2503
           1       0.83      0.83      0.83      1692

    accuracy                           0.86      4195
   macro avg       0.86      0.86      0.86      4195
weighted avg       0.86      0.86      0.86      4195
```



Interpretation of the classification report:

- **Precision**: Out of all predicted instances for a class, how many were actually that class? For example, out of all instances predicted as class 0, 88% were actually class 0. Similarly, for class 1, it's 83%.
- **Recall**: Out of all actual instances of a class, how many were correctly predicted? For example, the model identified 88% of all actual class 0 instances and 83% of all actual class 1 instances.
- **F1-score**: It's a balance between precision and recall. It considers both false positives and false negatives. Values closer to 1 are better. Here, it's 0.88 for class 0 and 0.83 for class 1.
- **Support**: The number of actual instances of each class in the test data.
- **Accuracy**: Overall, how many instances were correctly predicted? Here, it's 86%.
- **Macro avg**: The average of precision, recall, and F1-score for all classes, without considering class imbalance.
- **Weighted avg**: The average of precision, recall, and F1-score for all classes, weighted by the number of instances for each class.

Outcome:

The project achieved notable results in exploring and comparing the effectiveness of LDA and NMF for topic modeling. By incorporating these techniques along with sentiment analysis, you trained a logistic regression model that classified articles into the "Computer Science" category with an accuracy of 86%. This indicates a successful application of these topic modeling techniques in identifying and classifying textual data based on its content. The findings suggest that both LDA and NMF offer valuable insights into the underlying topics within a corpus of text, with each method providing its unique advantages in terms of topic density and interpretability. The project underscores the importance of feature engineering and model selection in the field of text classification and topic modeling.

**** END ****