CS 7641 Group 29 Machine Learning Project

Welcome to our CS 7641 group project repository!
We are working on detecting political bias in news articles

🎯 Project Overview

> This repository contains our comprehensive machine learning project for **CS 7641 - Machine Learning**. We are developing and implementing state-of-the-art ML algorithms to detect political bias in news articles.

Key Objectives

- Implement and compare multiple ML algorithms
- Conduct thorough data preprocessing and feature engineering
- Evaluate model performance using various metrics
- Provide insights on algorithm effectiveness and real-world applicability

---

📋 Project Components

Find the detailed contribution table and Gantt chart **here**

Component	Status	Description
📊 **Data Collection**	✅ Completed	Gathering and preparing dataset
🔧 **Preprocessing**	✅ Completed	Feature engineering and data cleaning
🤖 **Model Development**	🔄 In Progress	Implementing ML algorithms
📈 **Evaluation**	🔄 In Progress	Performance analysis and comparison
📝 **Documentation**	🔄 In Progress	Comprehensive project documentation

---

👥 Team Members

Name	Contact
Bhumika Chopra	bchopra9@gatech.edu
Carla Du Plessis	cplessis6@gatech.edu
Chaithanya Shyam Delanthabettu	csd3@gatech.edu
Nandan Bharatkumar Parikh	nparikh44@gatech.edu
Sonika Vuyyuru	svuyyuru7@gatech.edu

📚 Navigation

📖 Project Proposal & Report

View Proposal and Report

Complete project proposal and progress report including:

- Literature review and background
- Problem definition and motivation
- Detailed methodology and algorithms
- Expected results and metrics

💻 Repository

GitHub Repository

Source code and additional materials:

- Implementation files
- Dataset and preprocessing scripts
- Results and visualizations
- Documentation

🚀 Getting Started

1. **Review the proposal and progress report** to understand our methodology
2. **Check out our GitHub Repository** for source code

Last updated: 10/03/2025

CS 7641 Group 29 ML Project

Proposal & Midterm Report

Political bias detection in news articles

Introduction and Background

News media serve as a primary channel through which the public learns about politics. However, news articles often reflect subtle political bias through word choice, framing, selection, and source attribution. Detecting such bias is critical for improving transparency and helping readers contextualize coverage. Traditionally, political bias in news has been modeled as a three-way classification task—left, center, right—aligned with known ideological leanings of outlets. Early computational approaches relied on sentiment analysis and lexical statistics but often failed to capture more nuanced forms of bias, such as selective topic coverage or framing strategies [1].

Recent NLP advances have improved political bias detection by combining textual features, metadata, and embeddings [2]. Sentence-BERT [3] enables semantic comparison, while contrastive learning [5] helps capture subtle ideological framing. Social network–aware models like Retweet-BERT [5] enhance predictions by incorporating diffusion patterns. Large language models have also been applied to framing detection, revealing nuanced narrative biases and misinformation cues in headlines and article text [6], [7]. Most recently, Rönnback et al. [8] introduced a large-scale AI-powered bias detector that not only outperforms existing models but also provides interpretable explanations of why outlets are categorized in a particular way.

Dataset Description

We will utilize a combination of datasets to ensure robustness in our developed methods.

- **BASIL** [9]: Provides event-level stance annotations for articles from Fox (Right), NYT (Center) and HuffPost (left), enabling fine-grained framing analysis.
- **AllSides (Kaggle)** [10]: Large corpus of articles with outlet-level bias labels, for weak supervision and baseline model training
- **GDELT Project [11]**: Massive, real-time global database of news, for outlet-level aggregation to proxy coverage bias
- **Media Bias Fact Check (MBFC) [12]** : Outlet-level bias and factuality ratings across thousands of news domains.

Problem Definition

Problem Statement

Current data-driven approaches for political bias detection often focus on superficial indicators such as sensational headlines or biased wording. More subtle forms of bias, including selective topic coverage, underreporting, or framing differences across events, remain largely undetected. This motivates the need for models that not only classify articles by bias (left, center, right) but also provide interpretable insights into how bias manifests across outlets and events.

Motivation

Accurate political bias detection has applications in media literacy, fact-checking, and reducing information asymmetry. By combining article-level features, outlet-level aggregations, and advanced NLP models, we aim to capture both overt and subtle bias, going beyond simplistic classification.

Methods

Data Preprocessing Methods Proposed

1. Deduplication, Text Normalization, Tokenization & Stopword Handling, Lemmatization/Stemming, Handling Noise and Class Balance
2. **TF-IDF**: Idenitfy keywords that are dense within a group of articles but uncommon across all documents. Provides an initial signal for clustering articles by orientation.
3. **Contextual Embeddings**: BERT produced vector representations of articles that preserve semantic meaning, forming features for our main analysis.
4. **DAPT (Domain Adaptive Pre-training)**: Fine-tune BERT to adapt to political data to capture vocabulary and discourse nuances.

Machine Learning Algorithms/Models Proposed

Supervised

1. **Baseline**: Shallow Deep Neural Architecture: Shallow NN/LSTM with TF-IDF inputs. A softmax layer outputs probabilities for Left, Center, or Right.
2. **Realistic**: Fine-tuned RoBERTa, DoBERTa model trained on labeled political articles to directly predict orientation. Paper by Jiang mentions motivations for political ideology detection in news articles using BERT [5]

Unsupervised

1. **Baseline**: K-Means (K=3), GMM, Hierarchical Clustering, DBScan, Spectral Clustering
2. **SimCSE (Ambitious)**: Use contrastive learning for sentence embeddings to robust text representations, clustering semantically similar sentences and capturing fine-grained political bias cues with a triplet loss function

---

Midterm Implementations

Links to Google Colab Notebooks:

1. Unsupervised Learning
2. Data Pre-processing

Data Processesing Method Implemented

Our motivation was to gain insights into the data structure, quality, and potential biases before performing more complex analysis. Complex analysis included cleaning the data by filtering out bad data and short texts to clarify our bias distributions. We used methods of Text Embedding and TF-IDF in our data preprocessing. Understanding these distributions helped us to make decisions about further data cleaning and modeling. Our general preprocessing pipeline looked as follows:

1. Loading the Dataset

Process: We downloaded the AllSides dataset from a kagglehub library and then used pandas.read_csv to load the data into a DataFrame. This allowed us to process the data within a Google Colab environment using Python libraries like pandas. The relevant columns “Text” and “Bias” were selected. The categorical “Bias” labels were mapped to numerical integers (0-4)

for easier processing, with 0 = “highly conservative”, 1 = “moderately conservative”, 2 = “centrist / balanced”, 3 = “moderately liberal”, and 4 = “highly liberal”.

Result: Created a DataFrame with 8112 rows and 5 columns.

2. Initial Visualization and Analysis of Distribution

Process: We visualized the distribution of text lengths using histograms, and generated count plots using seaborn to show the frequency of different bias categories and sources. The distribution of the political bias labels was visualized using a bar plot. We also did some rudimentary binning to show the count of each type of bias category for each source.

Result: The data displayed a high count of very short text entries (Figure 1) and the distribution across bias categories and sources. The political bias labels bar plot showed an imbalance in the dataset with a higher number of “highly conservative” articles. (Figure 2) We visualized the spread of article sources in Figure 3. From our bin counting of each type of content for each source, we also observed that each source only produces one type of political bias content. (Figure 4)

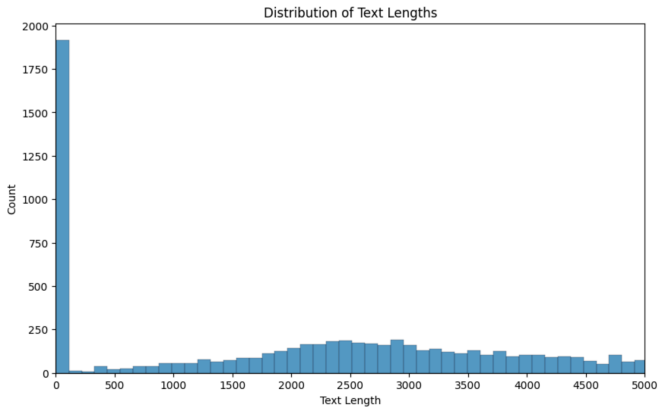

Figure 1: Distribution of text lengths

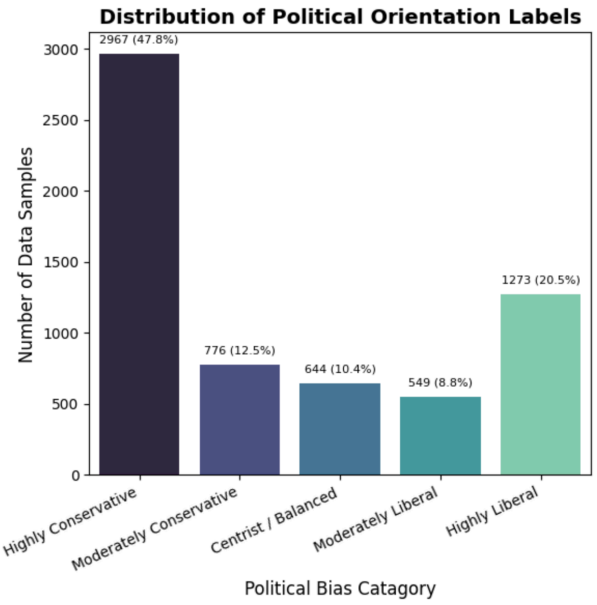

Figure 2: Dataset spread per category

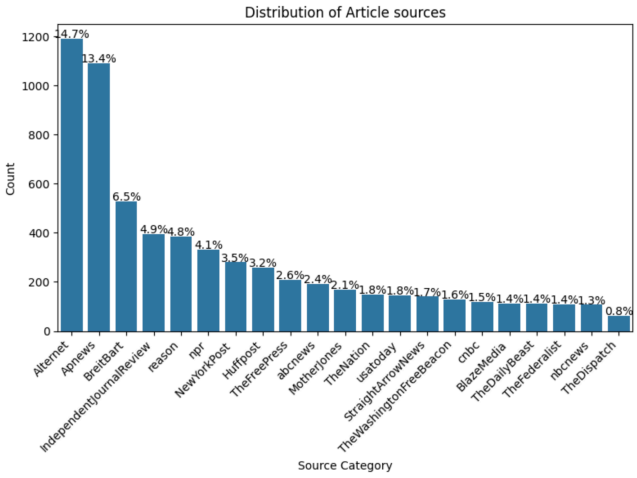

Figure 3: Spread per source

	Source	0
0	Alternet	[1190, 0, 0, 0, 0]
1	Apnews	[1091, 0, 0, 0, 0]
2	BlazeMedia	[0, 0, 0, 0, 111]
3	BreitBart	[0, 0, 0, 0, 529]
4	Huffpost	[257, 0, 0, 0, 0]
5	IndependentJournalReview	[0, 0, 0, 0, 395]
6	MotherJones	[169, 0, 0, 0, 0]
7	NewYorkPost	[0, 0, 0, 280, 0]
8	StraightArrowNews	[0, 0, 140, 0, 0]
9	TheDailyBeast	[111, 0, 0, 0, 0]
10	TheDispatch	[0, 0, 0, 61, 0]
11	TheFederalist	[0, 0, 0, 0, 110]
12	TheFreePress	[0, 0, 0, 208, 0]
13	TheNation	[149, 0, 0, 0, 0]
14	TheWashingtonFreeBeacon	[0, 0, 0, 0, 128]
15	abcnews	[0, 192, 0, 0, 0]
16	cnbc	[0, 0, 118, 0, 0]
17	nbcnews	[0, 107, 0, 0, 0]
18	npr	[0, 333, 0, 0, 0]

Figure 4: Count of each type of content per source

3. Filtering Out Bad Data:

To improve the quality of the dataset, we removed entries that are unlikely to be informative for text analysis, such as those with minimal text content.

Process: Created a new column text_length to store the length of the 'Text' column and then filtered the DataFrame to keep only rows where text_length was 50 or greater.

Result: This reduced the dataset to 6209 rows, focusing on more substantial articles.

4. Cleaning the Data:

To prepare the text data for natural language processing tasks by reducing noise, standardizing terms, and extracting meaningful components. This improves the effectiveness of techniques like TF-IDF vectorization and clustering.

Process: The ftfy library was used to fix text encoding issues. spaCy was employed to perform tokenization and lemmatization, converting words to their base forms and removing stop words.

Result: Standardized the text and extracted keywords for further analysis.

5. Normalization:

Textual data often consists of a lot of words that mean the same thing but are spelled differently, that is why we need to normalize the text. This is a way of reducing noise and improving the reliability of text analysis.

Process: Lowercase conversion, punctuation removal, contraction expansion, standardized spelling using SpaCY.

6. Lemmatization

Lemmatization is the process of reducing words to their base or dictionary form (lemma) while considering their context and part of speech. For example, “running”, “runs”, and “ran” are all reduced to “run”. Unlike stemming, which simply chops off word endings, lemmatization uses linguistic knowledge from a combination of dictionary lookups for irregular words (e.g., mice → mouse, was → be) and morphological rules based on each word’s part of speech to ensure the resulting word is valid and meaningful.

Process: We did this using SpaCy’s rule-based lemmatizer from the en_core_web_sm model.

Result: This process reduced vocabulary size and grouped different word forms under a single representative lemma, improving the quality of the TF-IDF representation used for modeling.

7. TF-IDF

This is a statistical measure used to represent text as numerical features, reflecting how important a word is within a document relative to the entire corpus. It combines two components: Term Frequency (TF) measuring how often a word appears in a document, and Inverse Document Frequency (IDF) which downweights words that appear frequently across many documents, reducing the influence of common terms like “the” or “is”. The resulting TF-IDF value is high for words that occur often in a document but rarely elsewhere, making them good indicators of the document’s content.

Process: TfidfVectorizer with a vocabulary limit of 5000 features was used to transform the lemmatized text into numeric vectors. This representation was then used for modeling and to extract the top-weighted words as keywords for each text sample.

8. Text Embedding:

Process: The **all-MiniLM-L6-v2** Sentence Transformer model was used to generate embeddings for the text data

Result: Resulting in a feature matrix X with a shape of (6209, 384).

Machine Learning Algorithms/Models Implemented

Unsupervised

We tried several unsupervised clustering methods, including K-means clustering, GMM, density-based methods like DBSCAN, hierarchical clustering, and spectral clustering. After data loading and preprocessing to generate our text embeddings, the steps we took for the unsupervised learning methods as follows:

1. Dimensionality Reduction:

Principal Component Analysis (PCA) was applied to reduce the dimensionality of the text embeddings from 384 to 50 components (X_pca). A 2-component PCA (X_vis) was also performed specifically for 2D visualizations.

2. Unsupervised Clustering Techniques:

We visualized the categorization of our unique political bias labels by setting our number of clusters to 5. We then evaluated these clustering methods by outputting visualizations of each which will be discussed in the results.

Several unsupervised clustering algorithms were initialized: - K-Means and Mini-Batch K-Means - Gaussian Mixture Model (GMM) and Bayesian Gaussian Mixture Model - Density-based methods: DBSCAN, HDBSCAN, OPTICS, and Mean Shift - Hierarchical Clustering: Agglomerative Clustering and Birch - Spectral Clustering

The current status of the models is that they have been fitted and their results visualized in 2D (and one attempt in 3D), which will be discussed below. The performance metrics have been calculated and are available in the computed_metrics DataFrame. Observations have been

made regarding the visualizations, and the potential impact of hyperparameters on DBSCAN and Spectral Clustering performance has been discussed.

Supervised

Learning Method 1

We are currently working on using the pre-processed TF-IDF vectors to do supervised learning. Our model architecture is a Shallow Neural Network with TF-IDF. Implementation: sklearn.neural_network.MLPClassifier with hidden layers (256, 128), Adam optimizer, early stopping.

Architecture: - Input: 5000 dimensional TF-IDF vectors - Hidden layers: [256, 128] with ReLU activation - Output: Softmax over bias classes (Left, Center, Right, etc.) - Regularization: Early stopping with validation monitoring

Why This Approach: This baseline establishes a performance floor using traditional NLP features. The shallow architecture with TF-IDF is computationally efficient and interpretable—we can directly examine which terms drive predictions. MLPClassifier's early stopping prevents overfitting on potentially noisy bias labels, and the moderate depth (2 hidden layers) balances expressiveness with generalization. This approach is expected to capture overt bias signals (explicit partisan vocabulary) but may struggle with subtle framing differences.

Expected Performance: Macro F1 ~0.65-0.75. TF-IDF excels at identifying explicit bias markers but lacks semantic understanding for nuanced framing.

Learning Method 2

Our next step is to try an advanced method which is Fine-Tuned RoBERTa. RoBERTa's pre-training on massive corpora provides rich contextual representations crucial for bias detection.

Key advantages include:
- Bidirectional Context: Captures how words interact within entire sentences, detecting framing through word ordering and syntax. - Transfer Learning: Pre-trained language understanding transfers to political domain with minimal fine-tuning - Attention Mechanisms: Self-attention layers learn which words/phrases carry ideological signals - Semantic Sensitivity: Distinguishes between "border security" vs "anti-immigrant policy"—same topic, different framing

The proposed architecture is:
- Base: roberta-base (125M parameters) - Fine-tuning: All layers with task-specific classification head - Training: 3 epochs, batch size 8, AdamW optimizer, learning rate warmup - Max sequence length: 512 tokens

Expected Performance: We expect this to perform better, with a macro F1 ~0.80-0.88 of around. Transformer models have shown strong results on similar framing detection tasks. The attention mechanism should identify subtle narrative differences across outlets covering the same events.

Results & Discussion

Project Goals

Our goal is to predict biases towards political ideologies by using text classifications and explainability to highlight certain words and tones that are prevalent in writing that is either left or right-leaning. We also think we can train our model by grouping data from specific news sources into the categories of either left or right-leaning, then by deploying the model on external news data, we can devise a metric to quantify the added bias to the article.

Expected Results

- Supervised classification (left/center/right) evaluated using accuracy and macro-F1.
- Unsupervised ECFD clusters evaluated with silhouette scores, Adjusted Rand Index, and human evaluation for interpretability.
- Insightful narratives showing selective coverage and framing differences across outlets.

Quantitative Metrics

1. **Likelihood/Probability Metrics**: Cross-Entropy Loss, Log Loss, Brier Score
2. **Performance Metrics**: Macro Averaged F1, ROC-AUC
3. **Explainability Metrics**: Shapley Additive Explanations (SHAP) Values, Integrated Gradients, LIME, Attention Visualization
4. **Clustering Metrics**: Silhouette Score, Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), Class Balance Checks, Expected Calibration Error (ECE)

Clustering Visualizations

K-Means and Mini-Batch K-Means clustering results: Visualized in 2D using the 2-component PCA reduced data, showing the cluster assignments and centroids. (Figure 1). The resulting clustering from K-Means shows overlapping clusters that are quite dense, indicating that there is some overlap between different political bias labels. In the future, we plan to reduce the density of the clustering and create more distinct clusters.

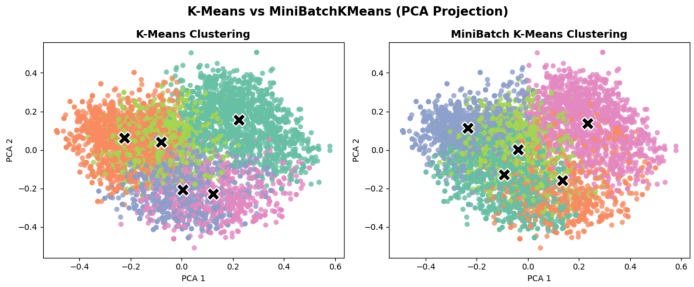

Figure 5: K-Means Clustering

Gaussian Mixture Model (GMM) Results: Visualized in 2D, including the cluster means and covariance ellipses.

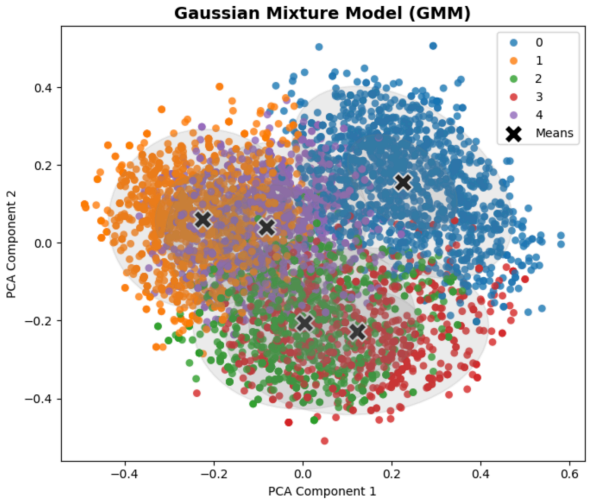

Figure 6: GMM Clustering

Hierarchical clustering (Agglomerative and Birch) results: visualized in 2D. (Figure 3) A dendrogram for hierarchical clustering was generated to show the merging of clusters. (Figure 4)

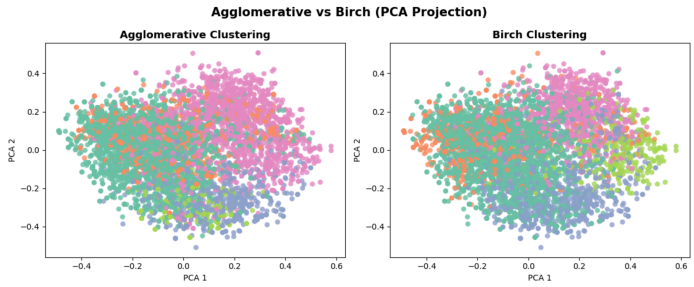

Figure 7: Agglomerative Clustering

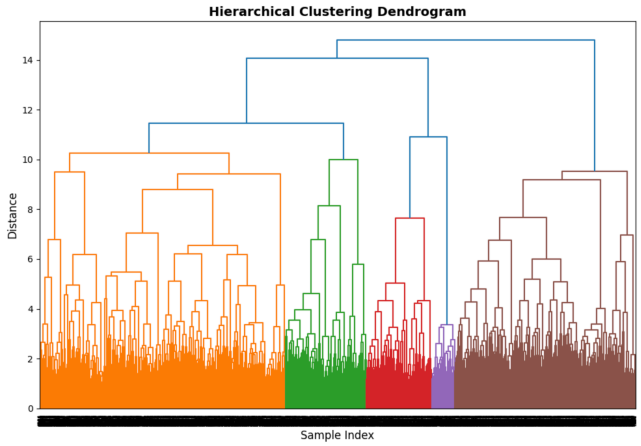

Figure 8: Dendogram

DBSCAN clustering results: visualized in 2D, and a 3D attempt was made. First, when we visualized all the points, we noticed that a majority of the points were identified as noise. (Figure 5) We then removed the noise points to just viusalize the DBSCAN clustering. (Figure 6)

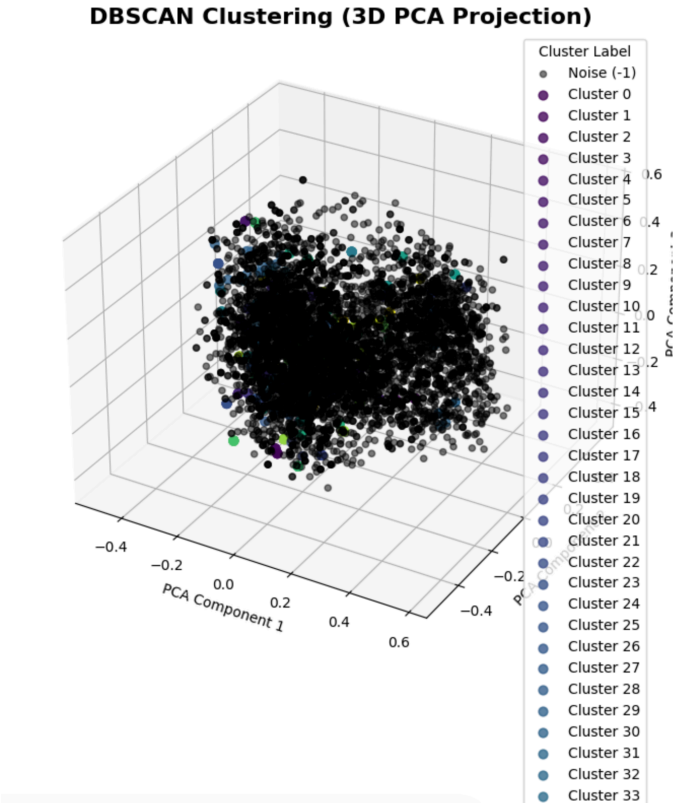

Figure 9. DBSCAN without noise

Figure 9: DBSCAN without noise

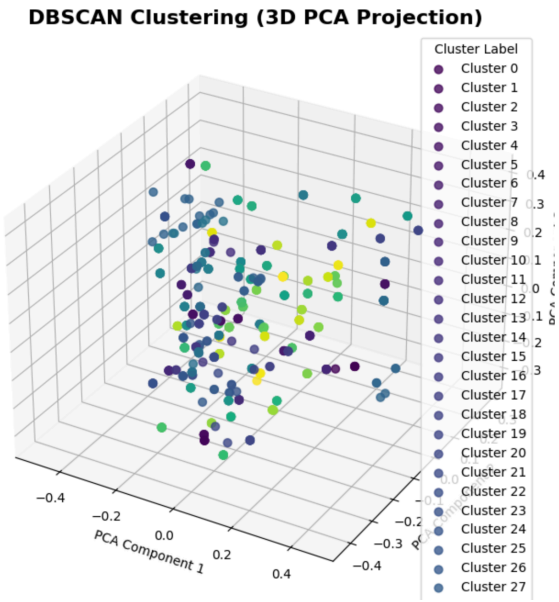

Figure 10: DBSCAN with noise

Spectral clustering results: visualized in 2D, and the n_neighbors hyperparameter was adjusted. (Figure 7)

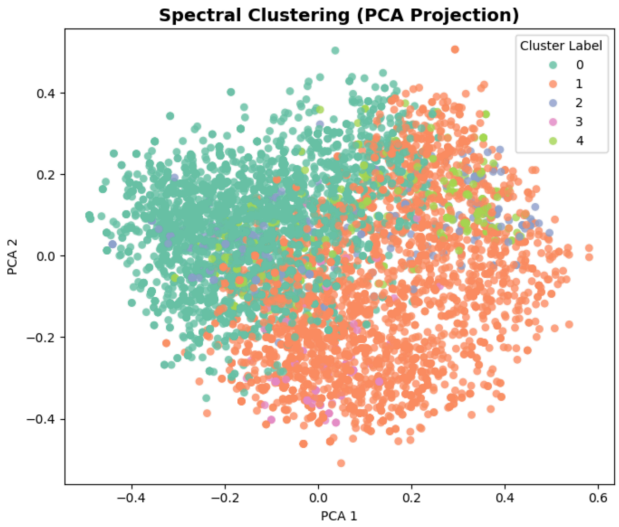

Figure 11: Spectral Clustering

Analysis of our Implemented Alogirthms/Model

We analyzed all our visualized clustering algorithms by conducting an evaluation of the metrics and normalizing the scores.

- A dictionary of clustering evaluation metrics was defined, including both intrinsic (Silhouette, Calinski-Harabasz, Davies-Bouldin) and extrinsic (Adjusted Rand Index, NMI, Homogeneity, Completeness, V-Measure) metrics.
- These metrics were computed for the labels obtained from fitting the various clustering algorithms.

Results

From the clustering evaluation metrics shown below, we can see that together with the results of the graph and the heatmap, DBSCAN, HDBSCAN, and Mean Shift had some of the highest overall performances across the quality metrics. They show to perform well in Calinski-Harabasz, Completeness, Davies-Bouldin, and Fowlkes-Mallows, with DBSCAN and HDBSCAN showing strong scores and handling cluster density well. We found that Agglomerative, GMM, and Birch had the lowest performance across metrics, showing poorer clustering for our bias data. To conclude, as we move forward, DBSCAN, HDBSCAN and Mean Shift are the best performing algorithms and will be prioritized in further modeling for our project.

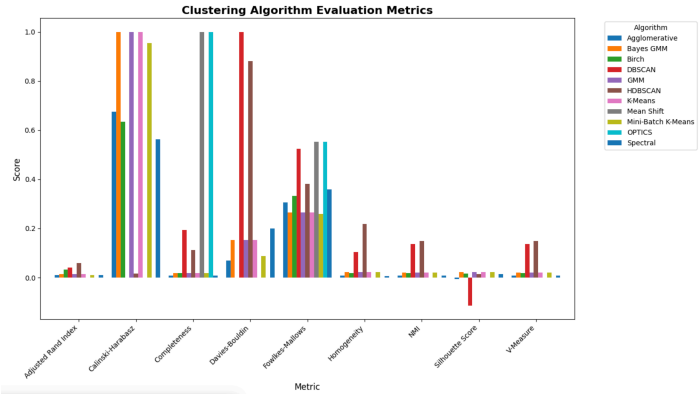

Figure 12: Clustering Evaluation Metrics

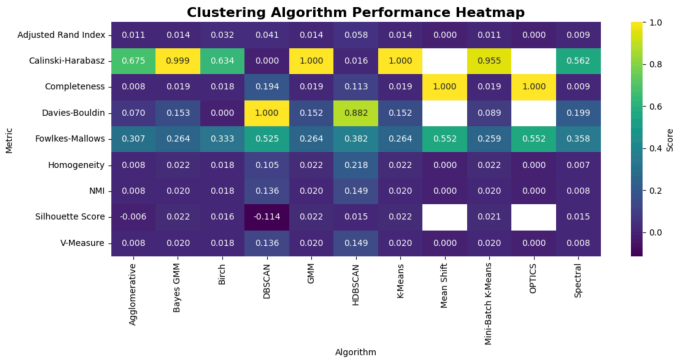

Figure 13: Clustering Evaluation Heatmap

Next Steps

- Visualize the remaining density-based clustering results (HDBSCAN, OPTICS, Mean Shift) if desired.
- Further explore hyperparameter tuning algorithms like DBSCAN and Spectral Clustering to potentially improve clustering results.
- Analyze the computed_metrics DataFrame to quantitatively compare the performance of the different clustering algorithms based on chosen metrics.

- Find ways to reduce the density of the clusters and fine tune the presentation of our clusters by removing noise points
- Based on the visualizations and metric evaluations, select the most promising unsupervised learning technique for this dataset.
- Implement our supervised learning techniques which are hallow Neural Network with TF-IDF and fine-tuning RoBERTa

---

Novelty Ideas

Our project introduces several novel components beyond standard bias classification:

1. **Event-Level Contrastive Learning**: Use contrastive learning on articles about the same event to detect nuanced narrative differences (e.g. triplet loss) across outlets.
2. **Explainable and interpretable predictions**: Using model interpretability tools, we provide reasoning for each classification, highlighting which textual features and coverage decisions influenced the bias label; focusing on tone, selection, percentage of fact vs. opinion, and size biases.
3. **Generative de-biasing**: Fine-tune an LLM to generate a de-biased version of an article.

References

[1] R. M. Entman, “Framing: Toward Clarification of a Fractured Paradigm,” Journal of Communication, vol. 43, no. 4, pp. 51–58, 1993.

[2] R. Baly, G. Karadzhov, D. Alexandrov, J. Glass, and P. Nakov, “Predicting Factuality of Reporting and Bias of News Media Sources,” in Proc. ACL, 2018.

[3] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in Proc. EMNLP-IJCNLP, 2019.

[4] P. Khosla et al., “Supervised Contrastive Learning,” in Proc. NeurIPS, 2020.

[5] J. Jiang, X. Ren, and E. Ferrara, “Retweet-BERT: Political Leaning Detection Using Language Features and Information Diffusion on Social Networks,” Proc. AAAI ICWSM, vol. 17, pp. 459–469, Jun. 2023. doi: 10.1609/icwsm.v17i1.22160.

[6] A. Pastorino et al., “Decoding News Narratives: A Critical Analysis of Large Language Models in Framing Detection,” 2024.

[7] Y. Wang, S. Frederick, Y. Duan et al., “Detecting Misinformation through Framing Theory: the Frame Element-based Model,” 2024.

[8] J. Rönnback, J. Carlsson, C. Calleja, and R. Feldman, “Automatic large-scale political bias detection of news outlets,” PLOS ONE, vol. 19, no. 8, e0321418, Aug. 2024. doi: 10.1371/journal.pone.0321418.

[9] L. Fan, M. White, E. Sharma, R. Su, P. K. Choubey, R. Huang, and L. Wang, “In plain sight: Media bias through the lens of factual reporting,” EMNLP, 2019. doi: 10.48550/arXiv.1909.02670

[10] S. Haldar, “AllSides : Ratings of bias in electronic media,” Kaggle.com, 2021. https://www.kaggle.com/datasets/supratimhaldar/allsides-ratings-of-bias-in-electronic-media/data

[11] The GDELT Project, “The GDELT Project,” Kaggle.com, 2015. https://www.kaggle.com/datasets/gdelt/gdelt

[12] idiap, “GitHub - idiap/Factual-Reporting-and-Political-Bias-Web-Interactions: Mapping the Media Landscape: Predicting Factual Reporting and Political Bias,” GitHub, 2024. https://github.com/idiap/Factual-Reporting-and-Political-Bias-Web-Interactions

---

Checklist Progress

✅ Completed

- Literature Review
- Dataset Description
- Dataset Link (if applicable)
- Problem Definition
- Motivation
- 3+ Data Preprocessing Methods Identified
- 3+ ML Algorithms/Models Identified
- CS 7641: Unsupervised and Supervised Learning Methods Identified
- CS 4641: Supervised or Unsupervised Learning Methods Identified
- 3+ Quantitative Metrics
- Project Goals (including sustainability and ethical considerations)
- Expected results
- 3+ References (preferably peer reviewed)
- 1+ In-Text Citation Per Reference

CS 7641 Group 29 ML Project