BBC News Articles: Comparison of Topic Modeling and Text Clustering Techniques

AGENDA

- Motivation
- Data Introduction
- Pre processing
- EDA
- LDA - Knime Workflow experiment
- Topic Modelling and Text Clustering Models/ Results
- Conclusion

Motivation

- Rapid growth of digital news content creates challenges in organizing and managing information effectively.
- A recent study explored how topic modeling and clustering methods perform on short, health-related texts like tweets and emails.
- It evaluated algorithms such as LDA, BTM, GSDMM, and KMeans (with TF-IDF and Doc2Vec) using both internal and external metrics.
- Results showed that no single method was universally superior; performance depended on the dataset and evaluation criteria.
- Notably, GSDMM and Online LDA produced coherent clusters, while LSI and KMeans aligned better with known categories.
- Motivated by this study, this project applies the same comparison framework to BBC news articles to see if similar patterns emerge in longer texts.
- It evaluates both topic modeling and clustering techniques using accuracy, precision, recall, and F1 score to understand which method best captures true news categories.

DATA

- Format: Collection of individual .txt files (one per article)
- Processing:
  - Used re and glob libraries in Python
  - Parsed and combined into a single DataFrame
- DataFrame Columns:
  - Title – Article title
  - Description – Main content
  - Category – Original category/topic

	count
Category	
Sport	511
Business	510
Politics	417
Tech	401
Entertainment	386

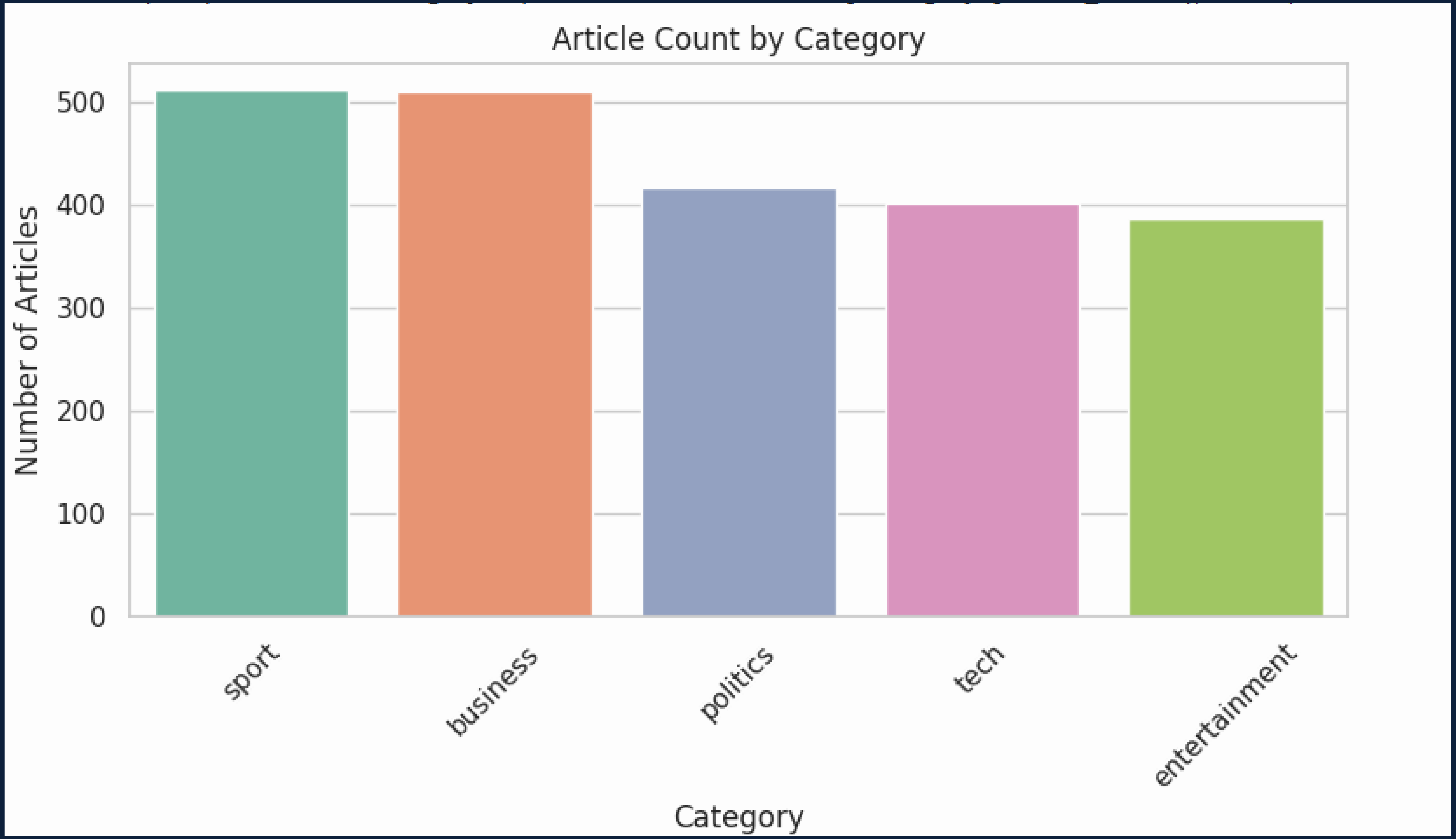
Pre processing

Before applying the topic modeling algorithms, the textual data underwent preprocessing, which included expanding contractions, removing punctuation, digits, extra whitespaces, and stop words. The remaining words were then lemmatized. After preprocessing, the corpus was vectorized using both Count Vectorizer and TFIDF Vectorizer, where each row represented a document, and each column corresponded to a unique term in the corpus.

EDA

```
(2225, 2)
category
sport            511
business         510
politics         417
tech             401
entertainment    386
Name: count, dtype: int64
```

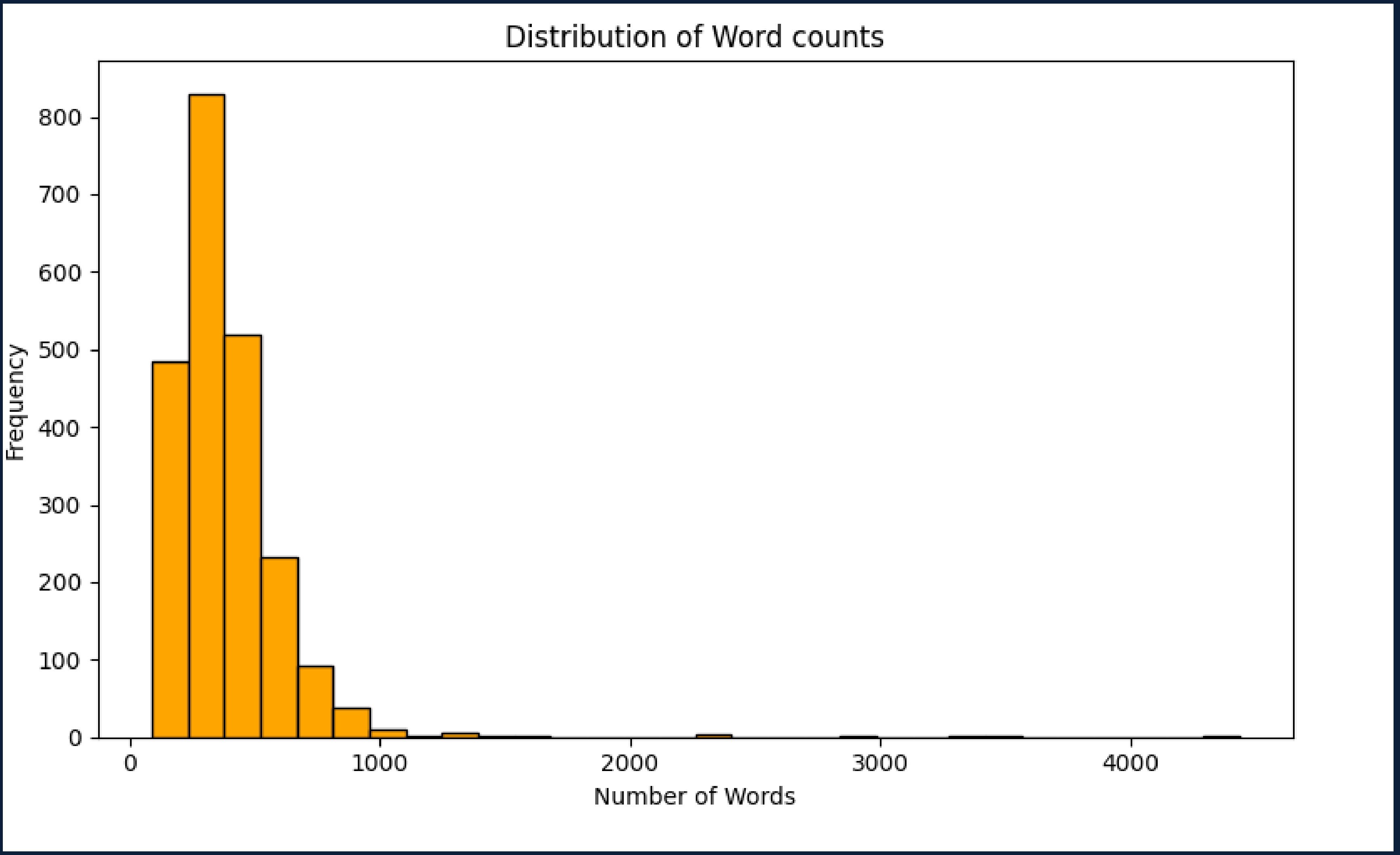
We used both Matplotlib & Seaborn



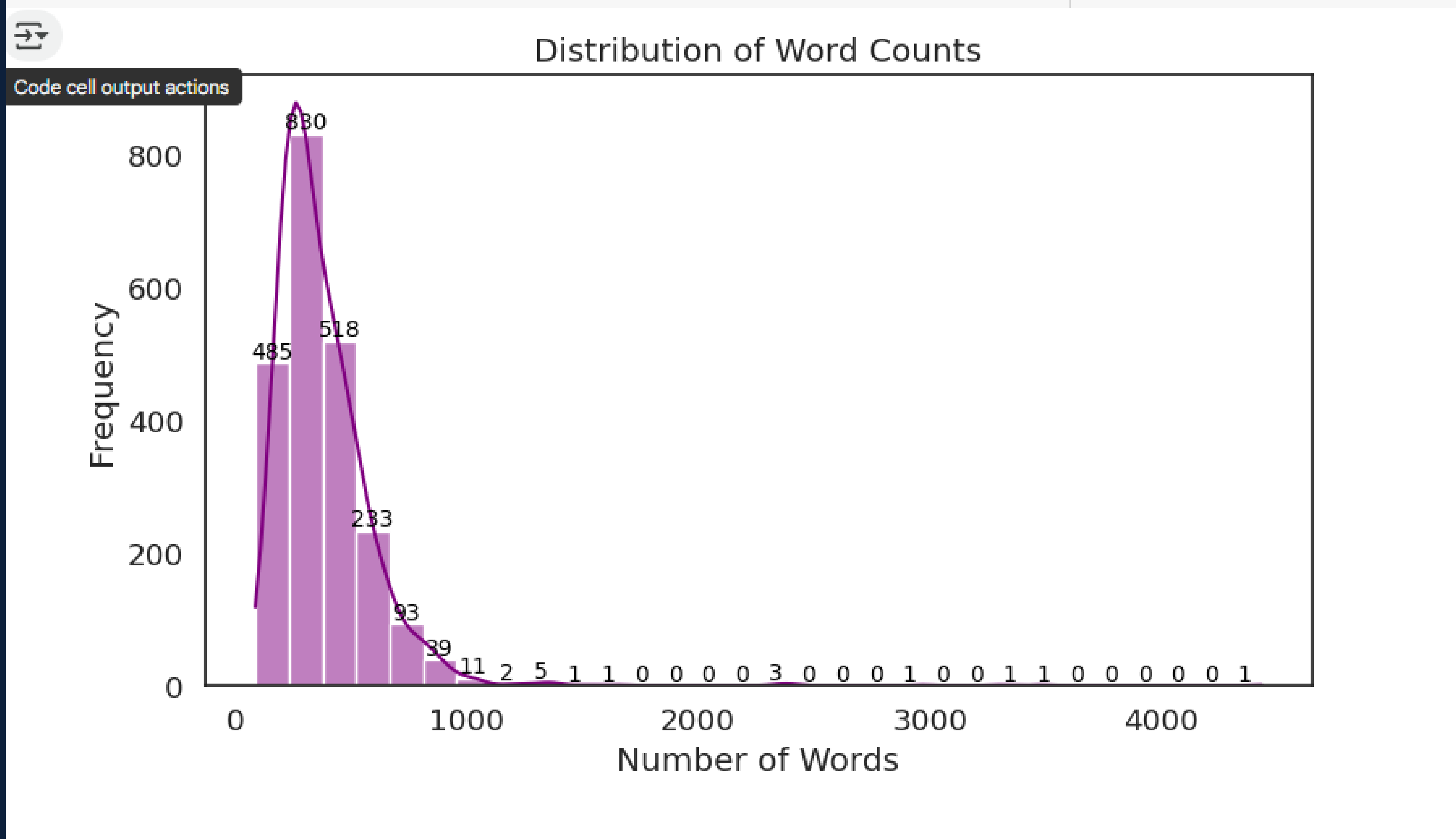
EDA (cont.)

Seaborn gives a prettier, clearer, and more insightful chart with less effort.

Matplotlib



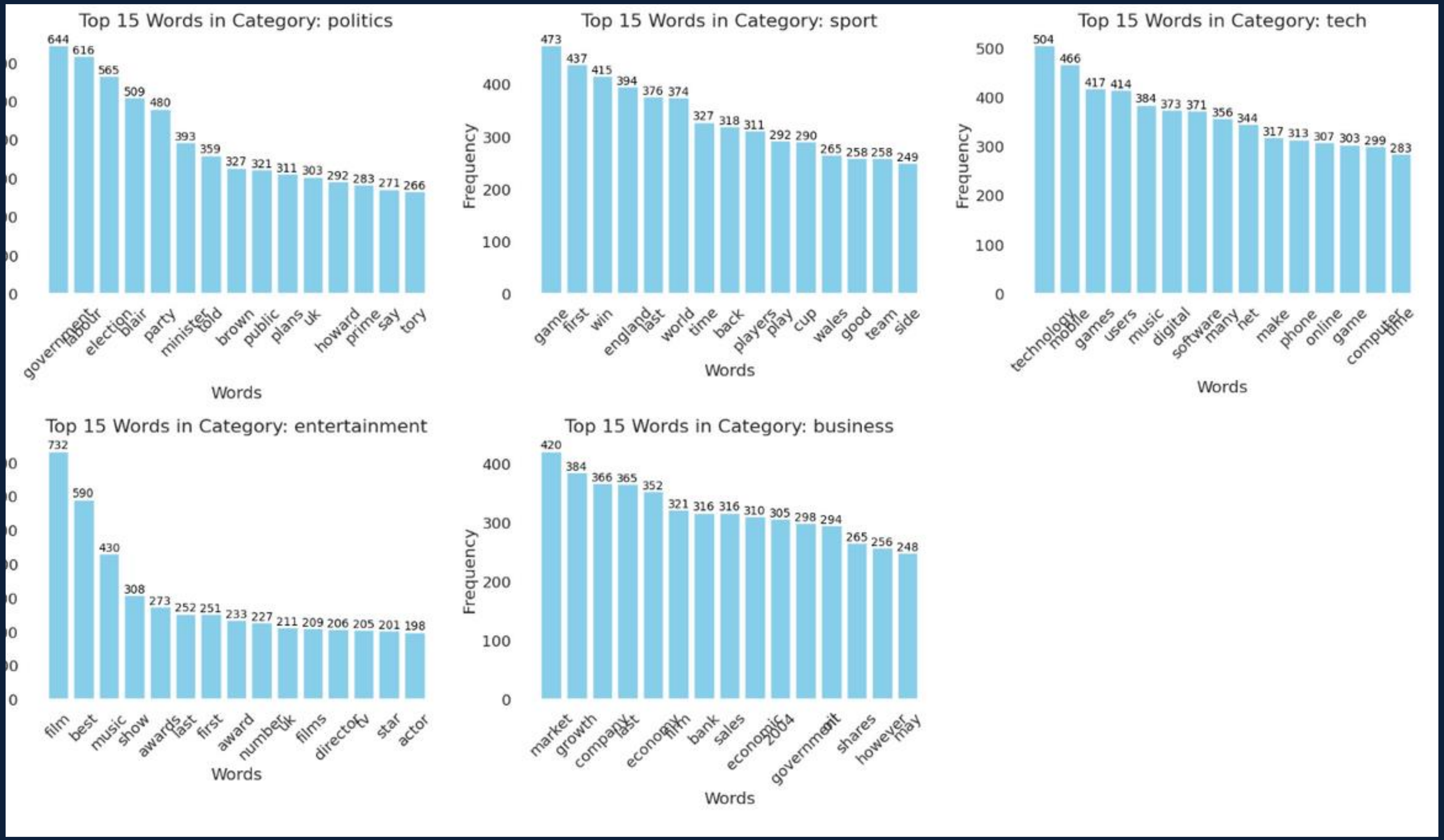
Seaborn



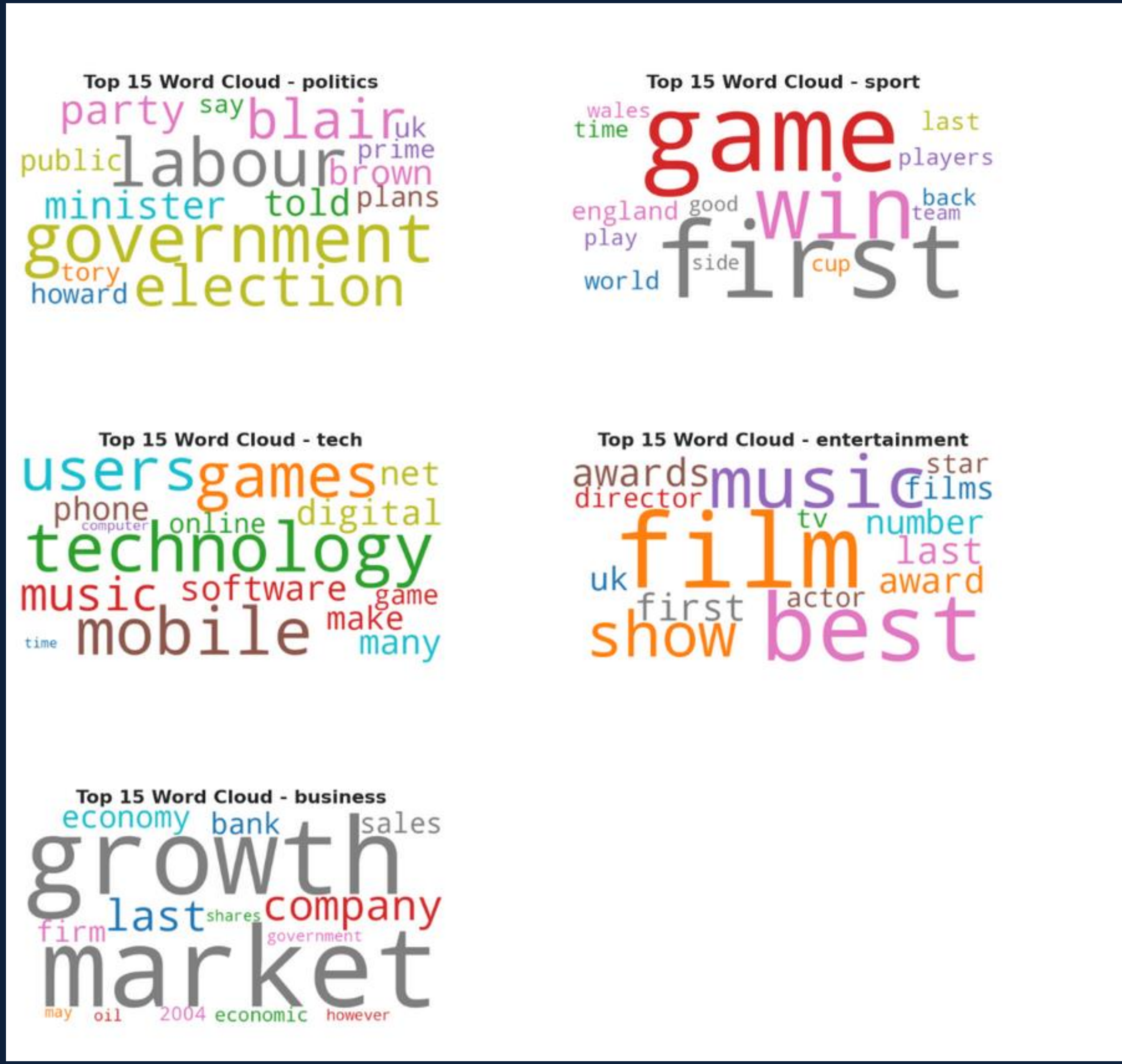
EDA (cont.)

Top 15 Frequent Words in each Category

Barchart of frequency

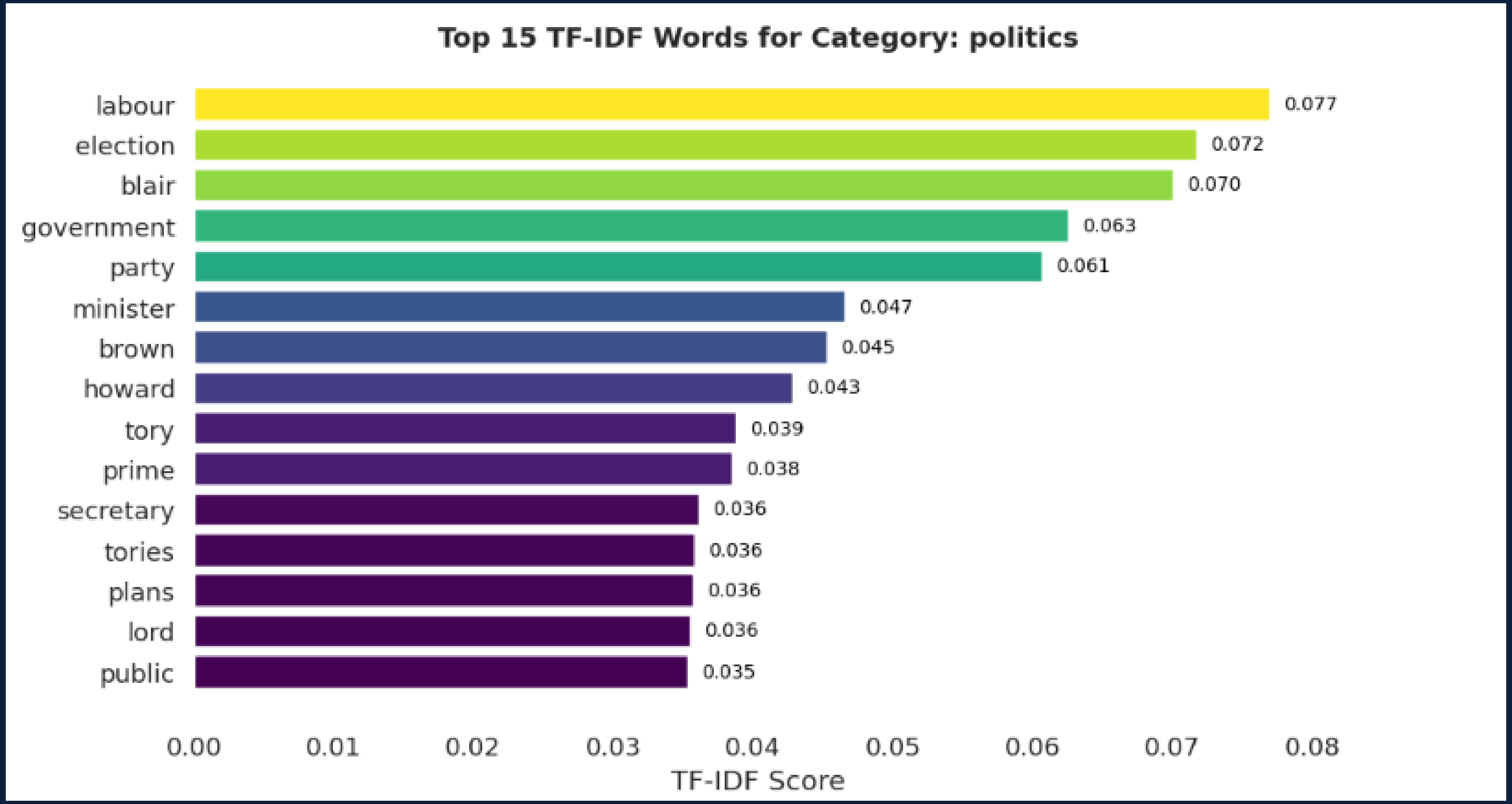


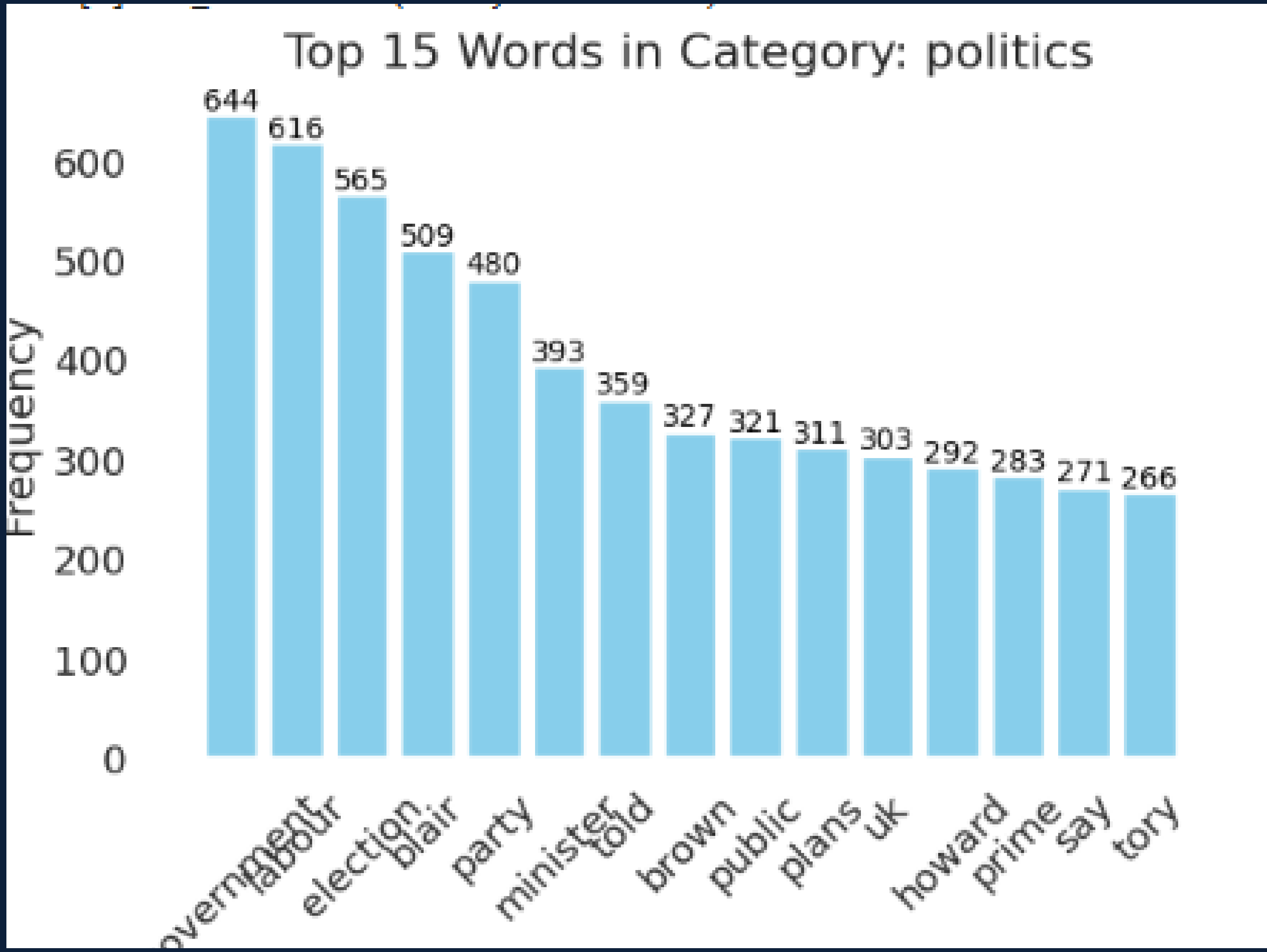
WordCloud



EDA (cont.)

TF-IDF Based Top 15 Words Per Topic (How distinctive the top word)



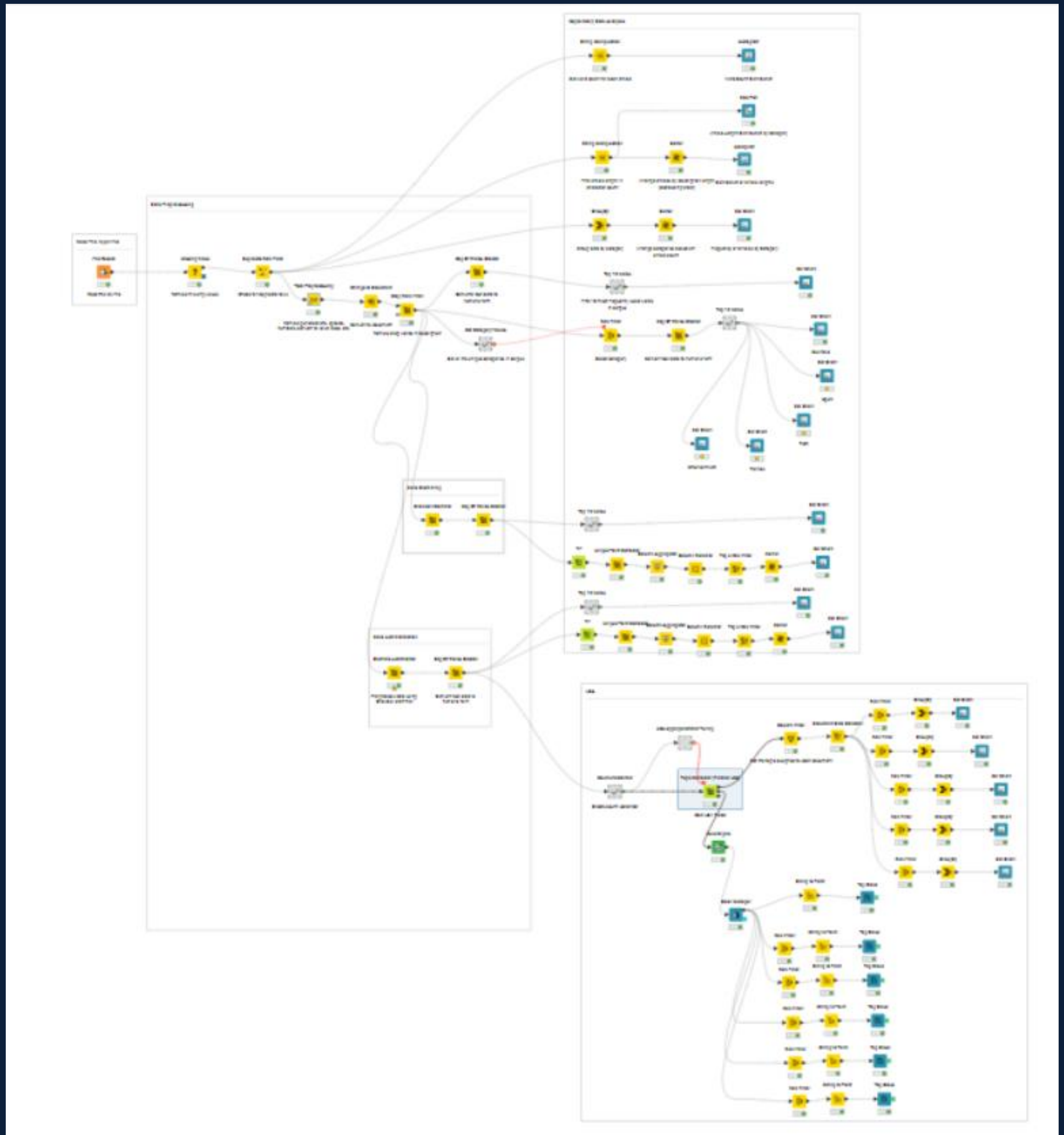


The TF-IDF chart provides a better insight into what defines for example the category ("Politics) because it filters out common words and focuses on what makes the content unique. While the frequency chart is easier to understand, it may include generic or overly common terms.

KNIME

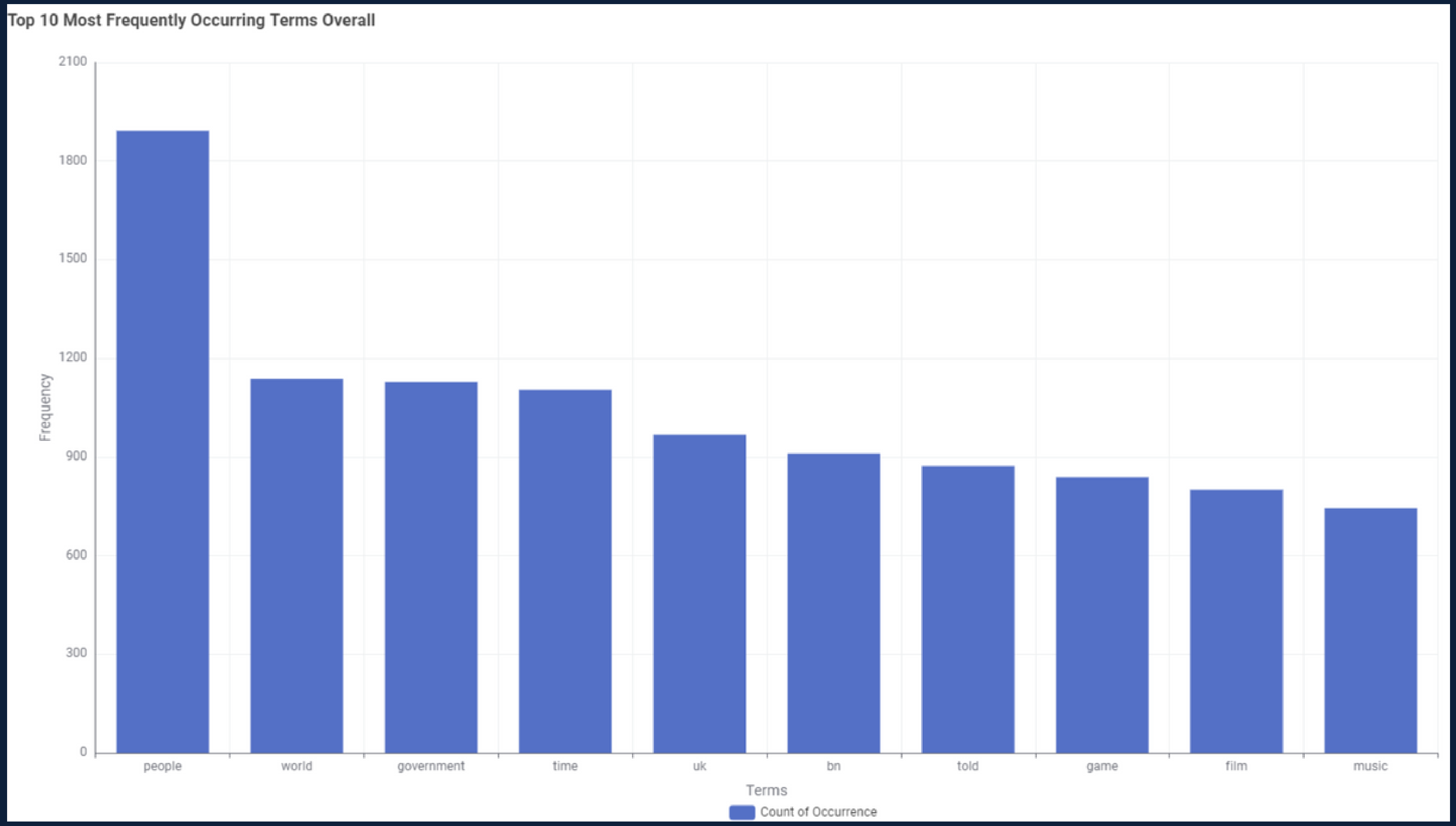
Data Analysis And Topic Modeling Using LDA

KNIME WORKFLOW



EDA

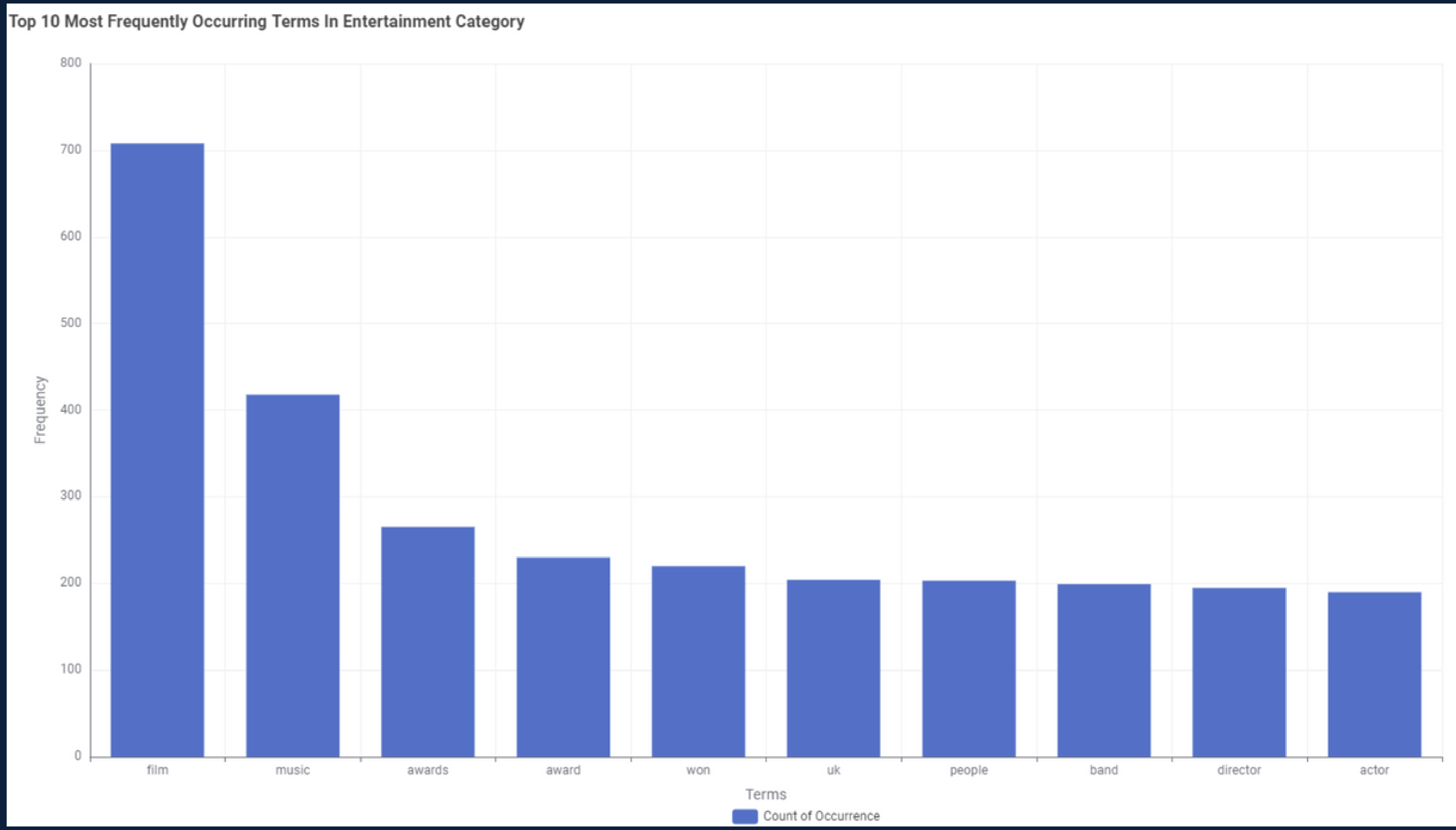
Most Frequent Terms Overall



EDA

Most Frequent Terms By Category

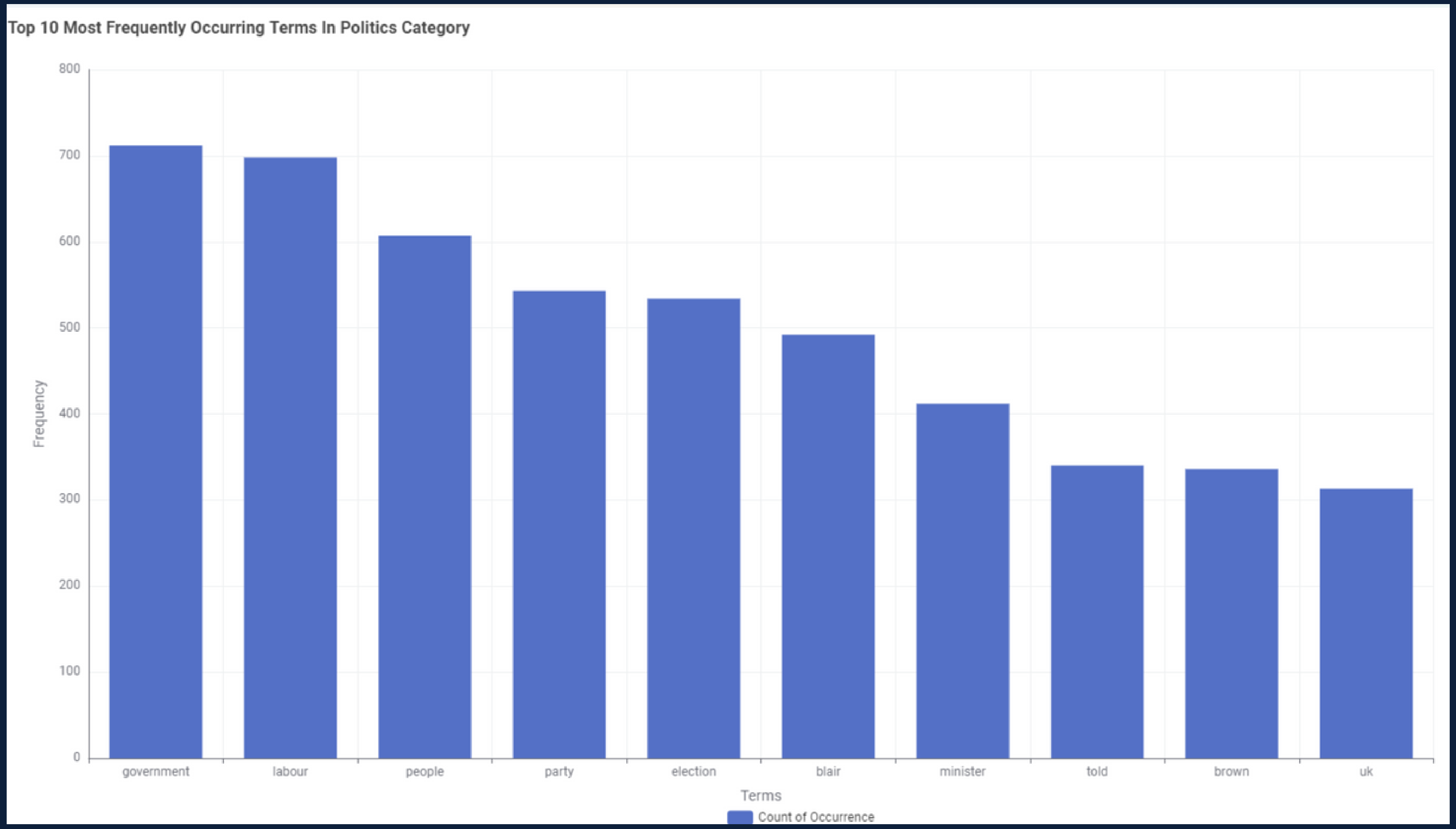
Entertainment



EDA

Most Frequent Terms By Category

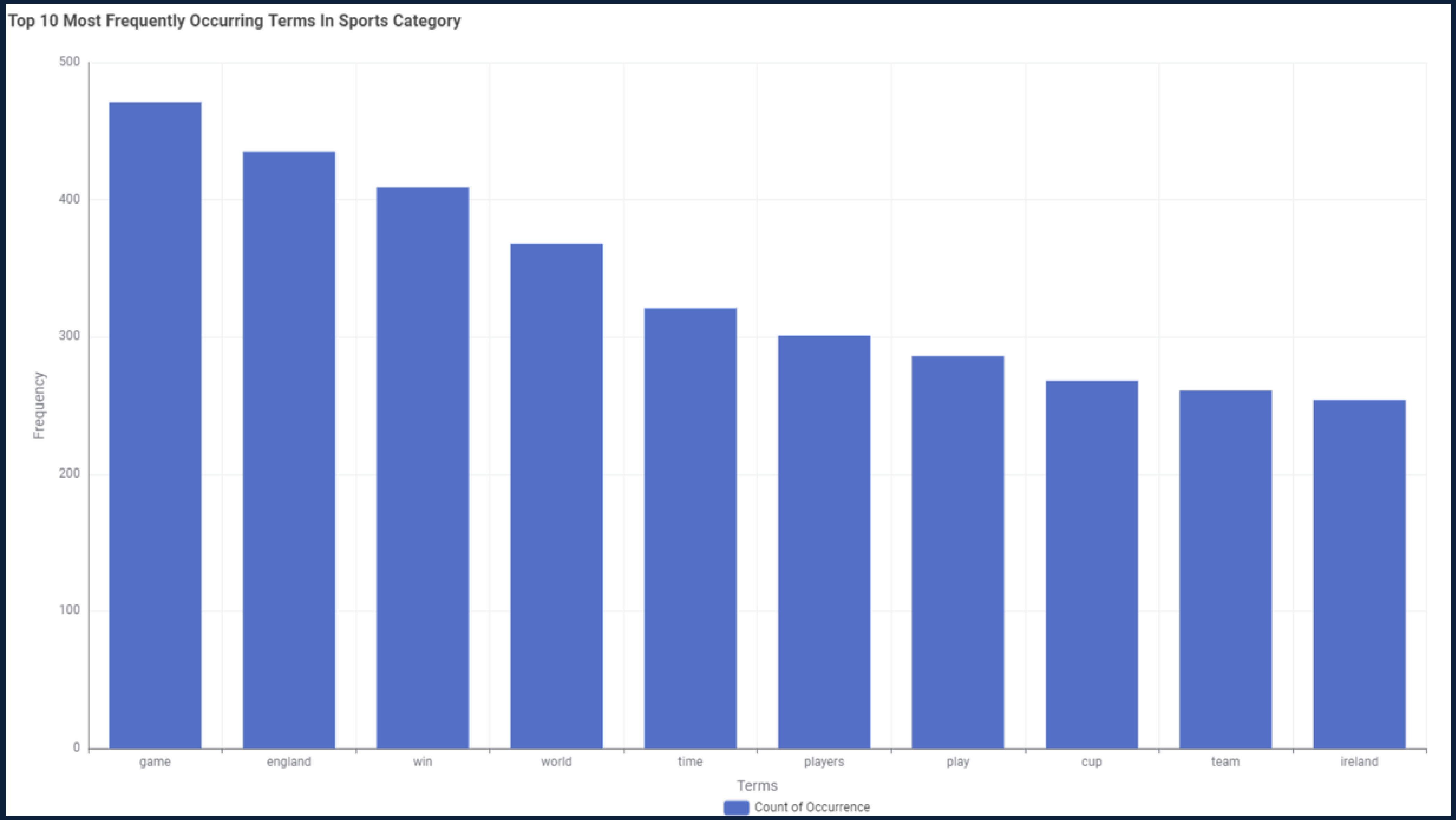
Politics



EDA

Most Frequent Terms By Category

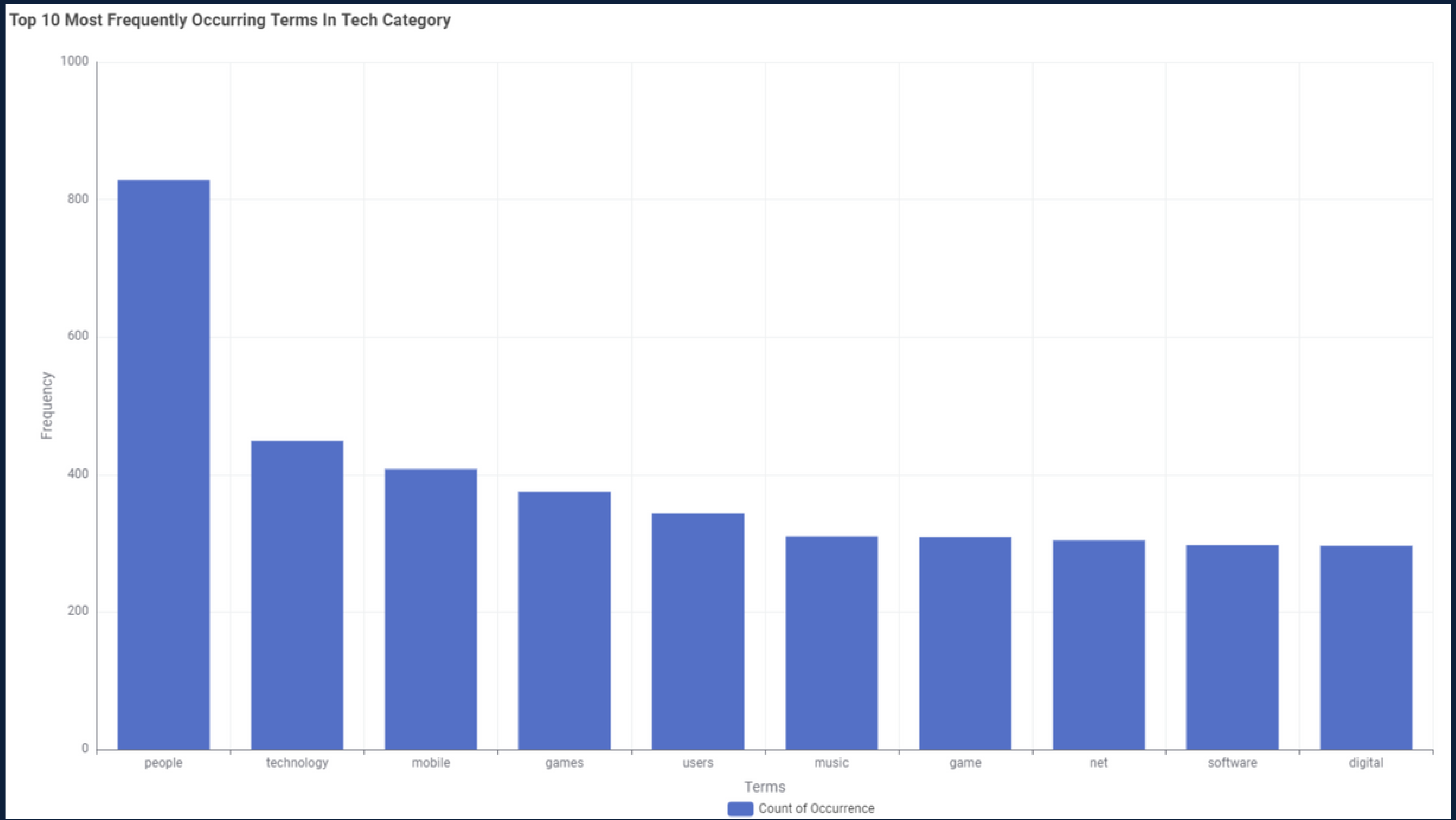
Sports



EDA

Most Frequent Terms By Category

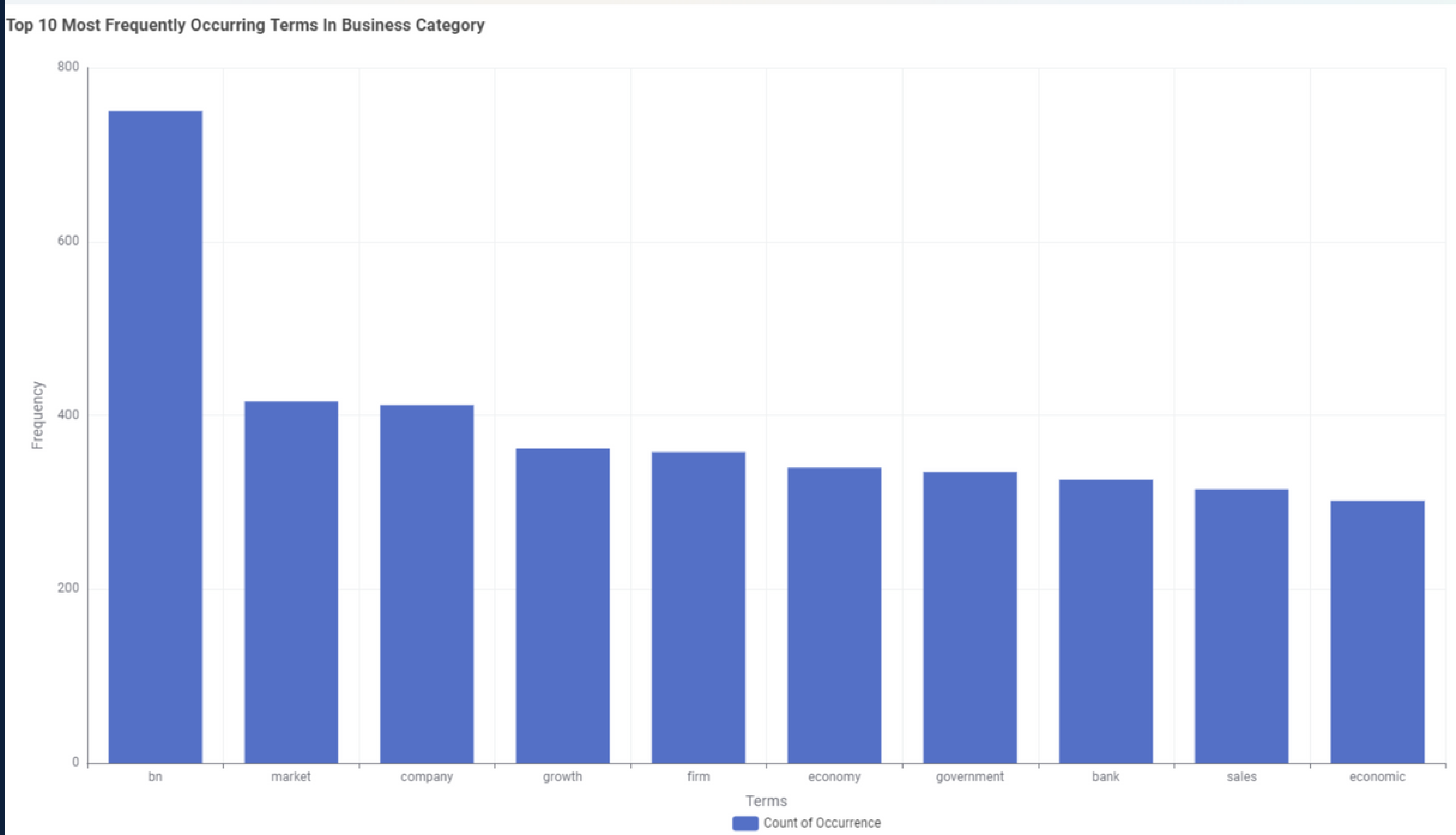
Tech



EDA

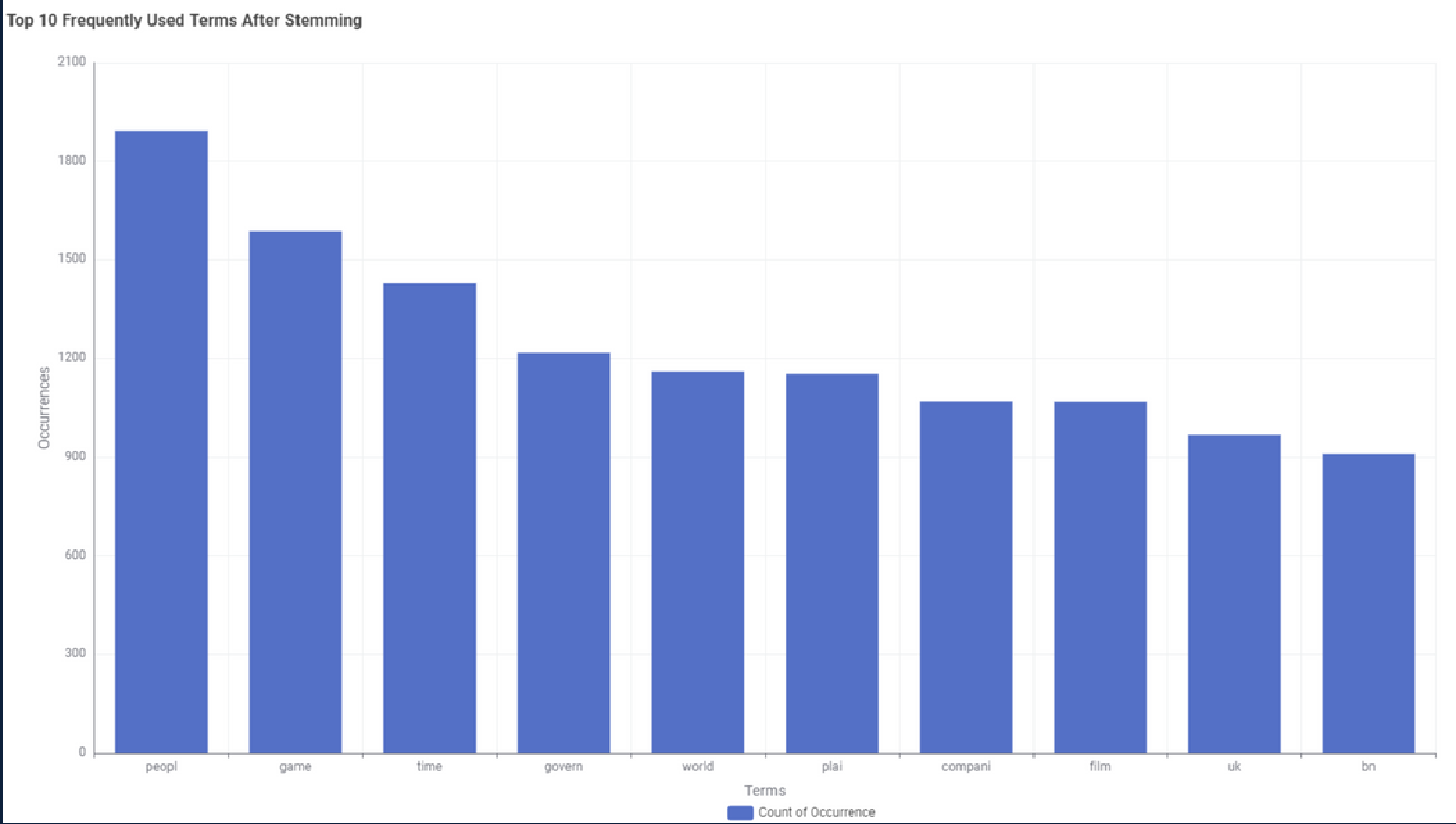
Most Frequent Terms By Category

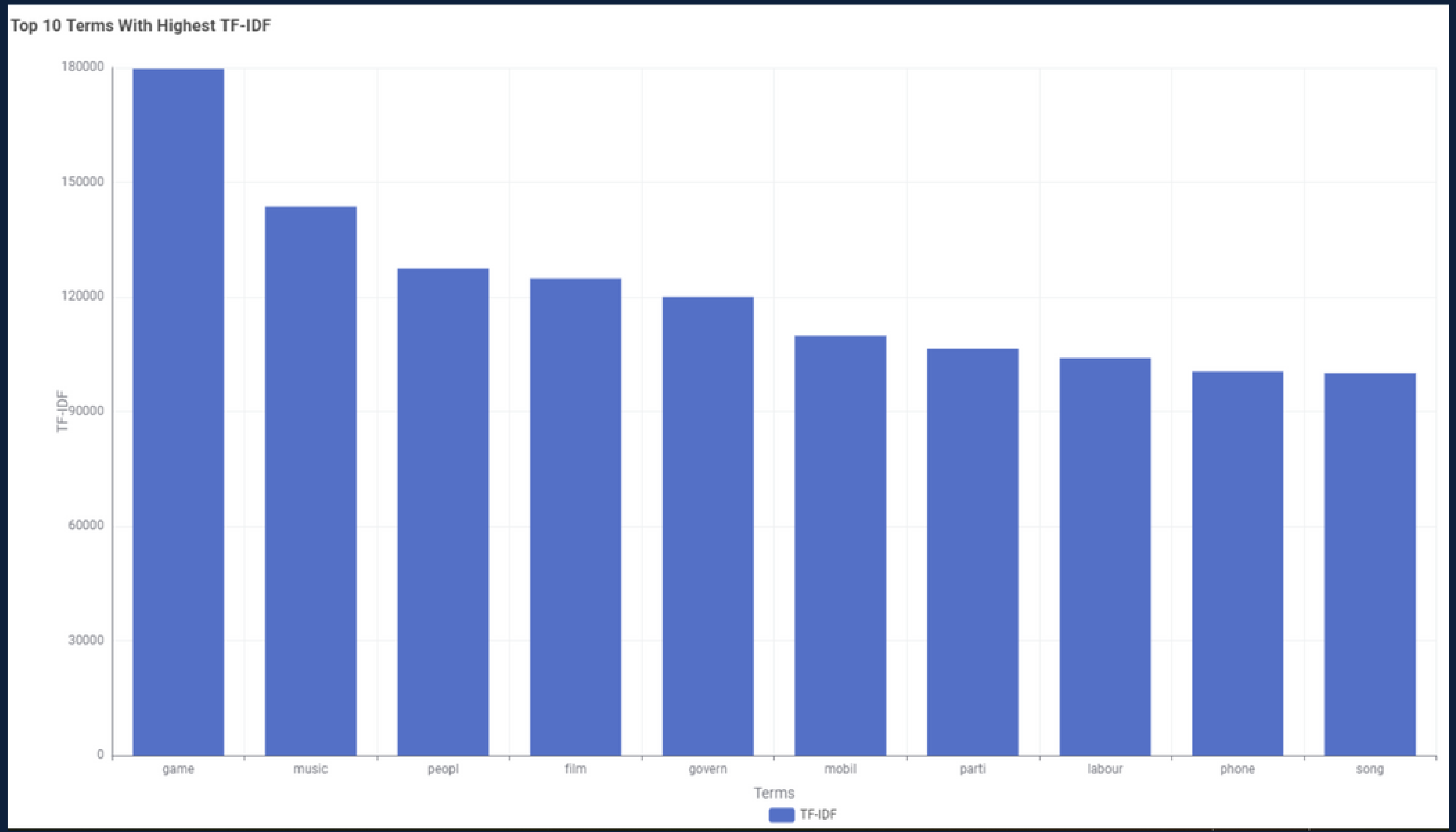
Business



EDA

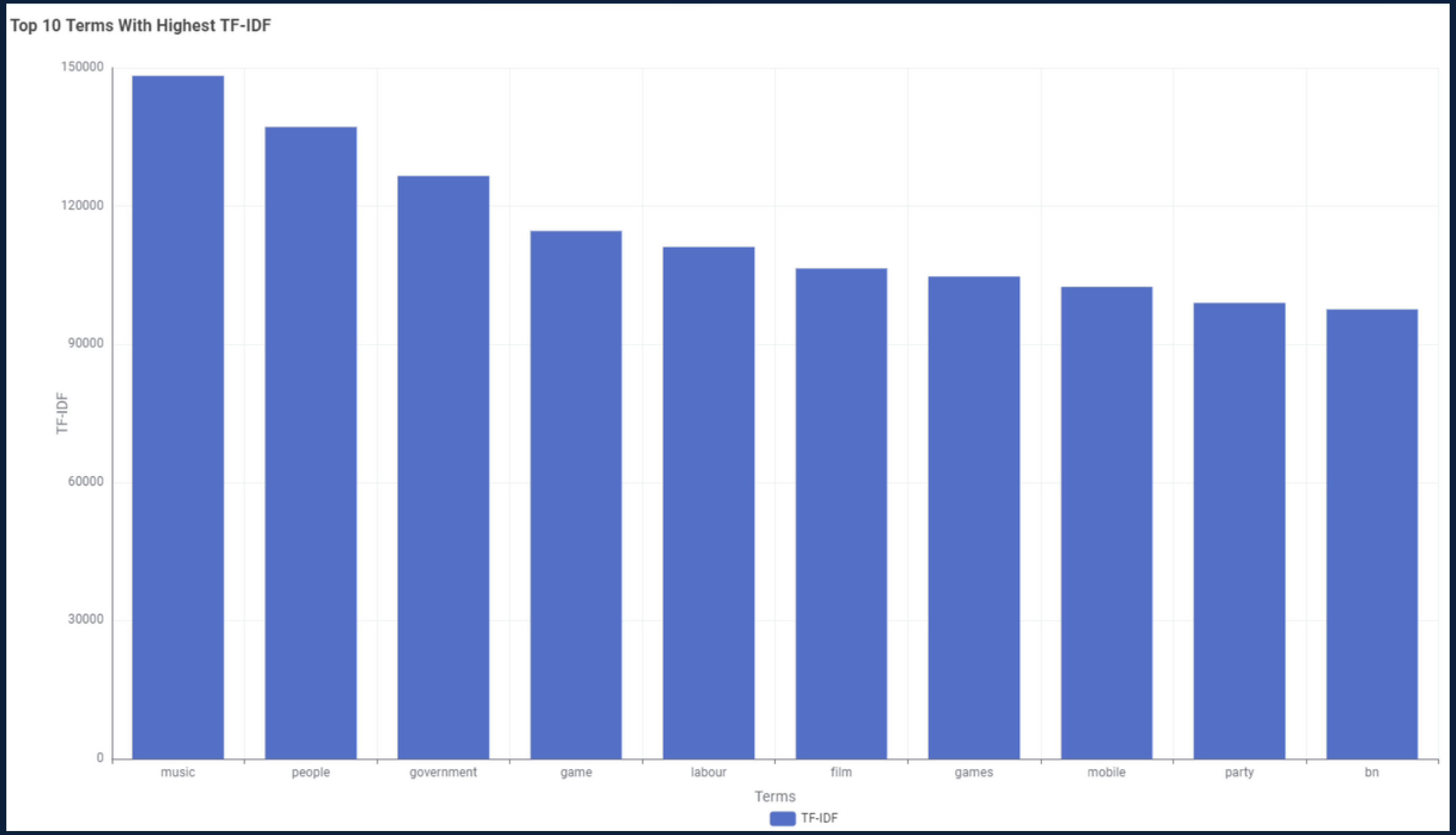
Data Stemming





EDA

Data Lemmatization



EDA

Possible Reasons For Difference Between KNIME and Python

Different tokenization methods

The tokenizer you select in KNIME can significantly impact results. The "String to Document" node offers different tokenizers (like "OpenNLP English WordTokenizer" vs. "OpenNLP SimpleTokenizer") that split text differently.

Preprocessing differences

Small variations in how you handle case conversion, punctuation removal, or stop word filtering can lead to different word counts.

Implementation variations

KNIME's text processing nodes and Python libraries like scikit-learn might implement algorithms slightly differently

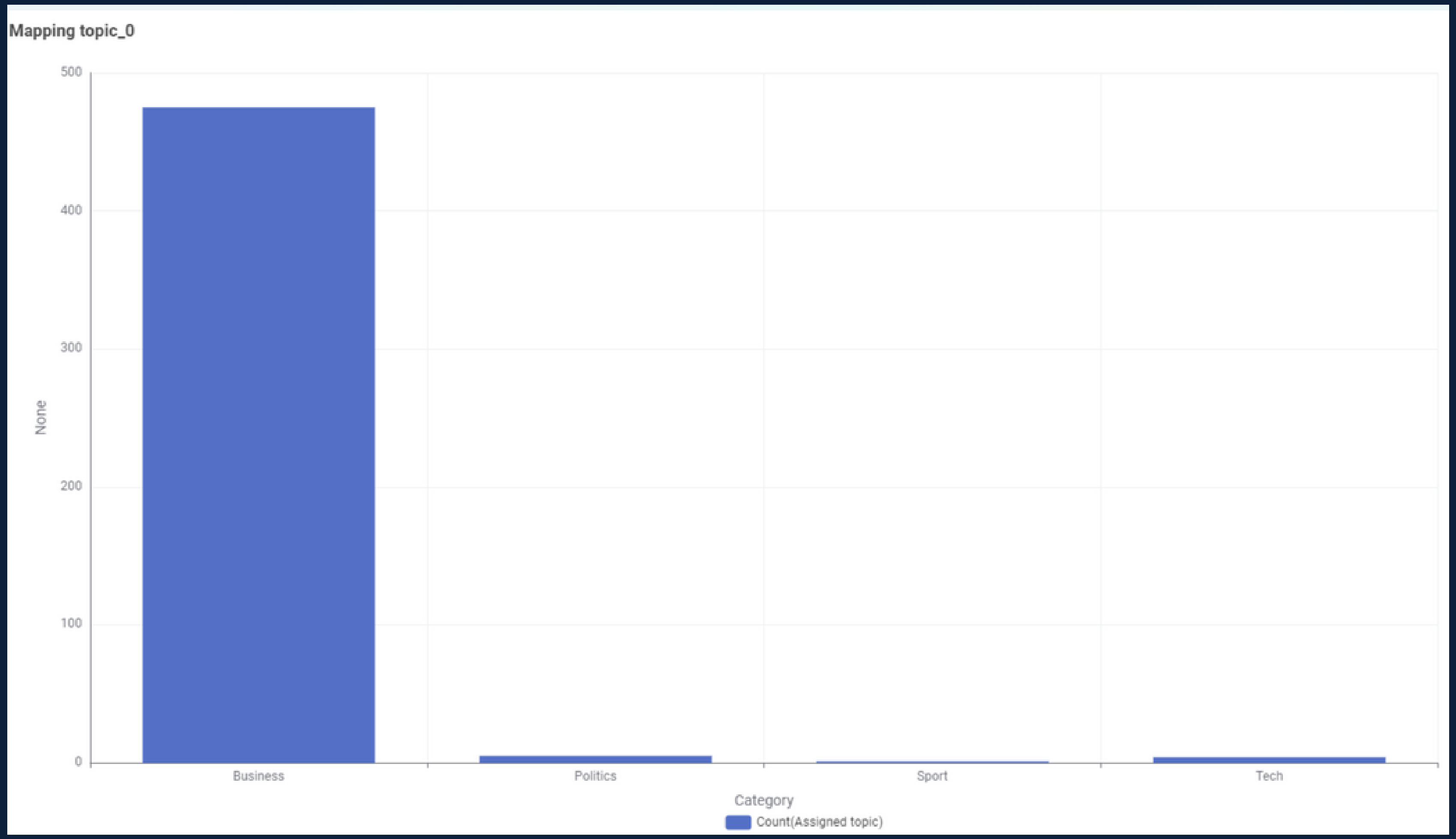
Distribution of Categories for Each Topic ID

topic_0

1: Group table   Flow Variables

Rows: 4 | Columns: 2   Table   Statistics

#	RowID	Category (String)	Count(Assigned topic) (Number (integer))
1	Row0	Business	475
2	Row1	Politics	5
3	Row2	Sport	1
4	Row3	Tech	4



We can say topic_0 is assigned to mostly Business articles

Distribution of Categories for Each Topic ID

topic_1

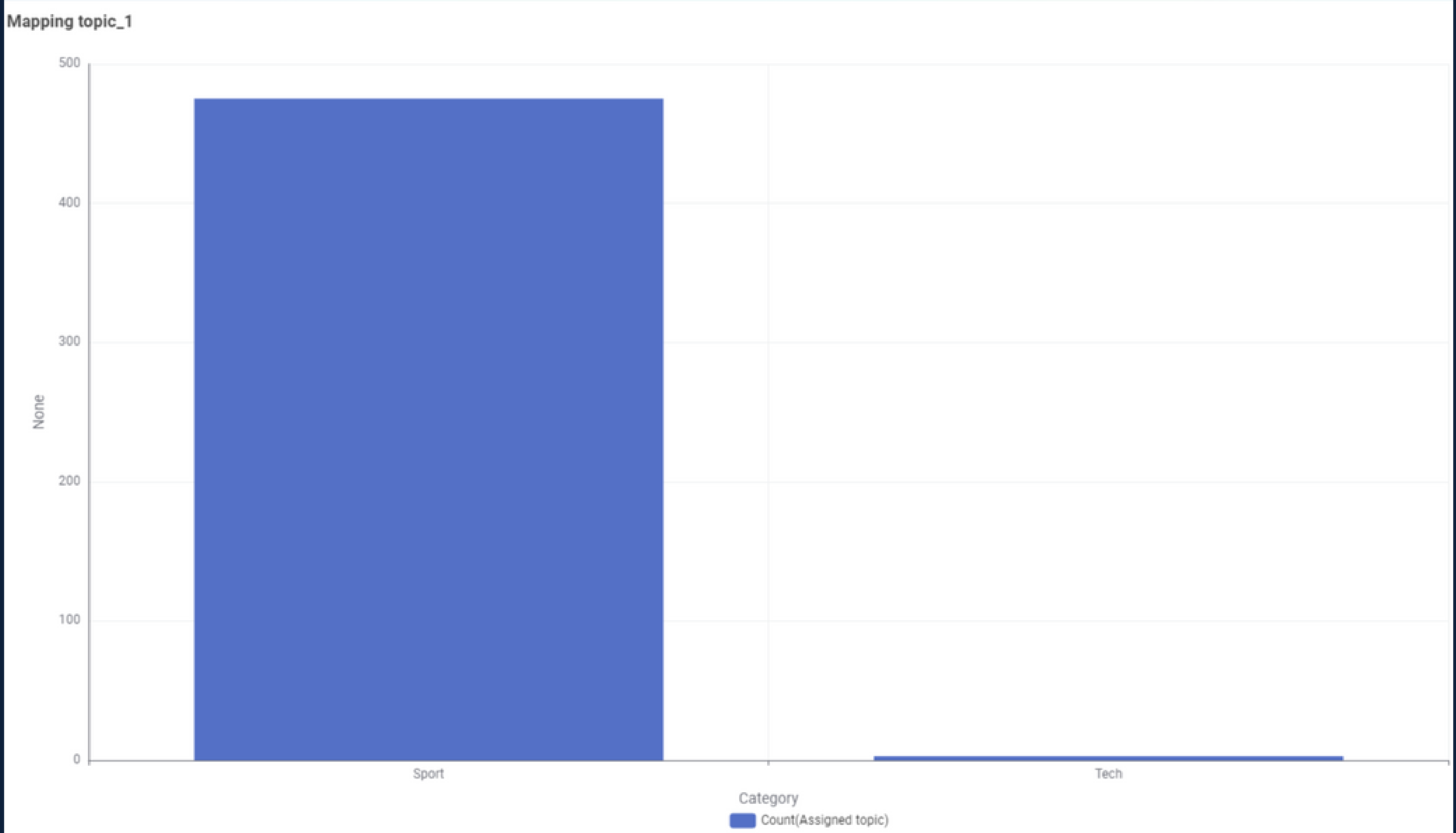
1: Group table | Flow Variables

Rows: 2 | Columns: 2

Table | Statistics

#	RowID	Category (String)	Count(Assigned topic) (Number (integer))
1	Row0	Sport	475
2	Row1	Tech	3

We can say topic_1 is assigned to mostly Sport articles



Distribution of Categories for Each Topic ID

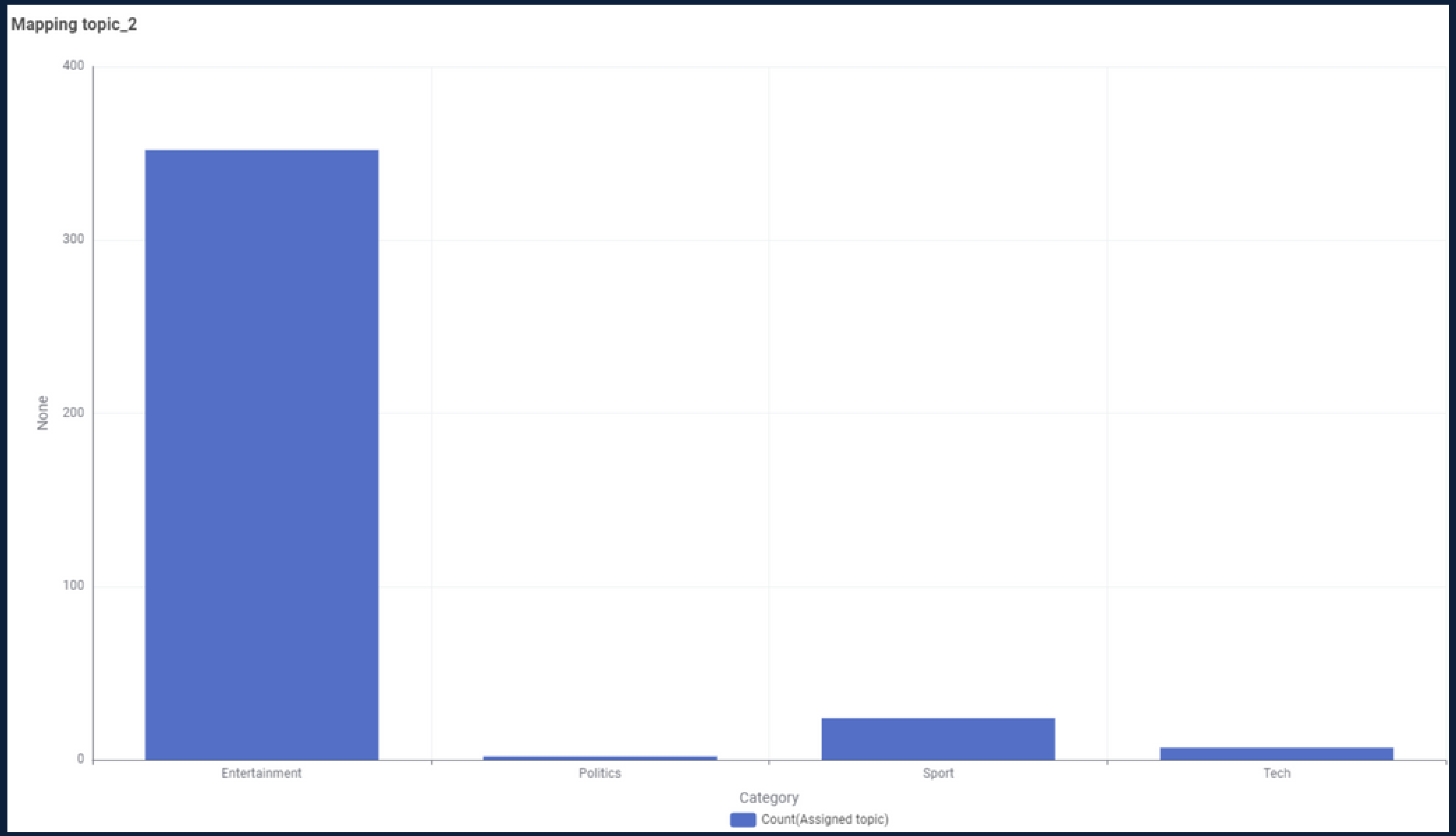
topic_2

1: Group table | Flow Variables

Rows: 4 | Columns: 2

Table | Statistics

#	RowID	Category (String)	Count(Assigned topic) (Number (integer))
1	Row0	Entertainment	352
2	Row1	Politics	2
3	Row2	Sport	24
4	Row3	Tech	7



We can say topic_2 is assigned to mostly Entertainment articles

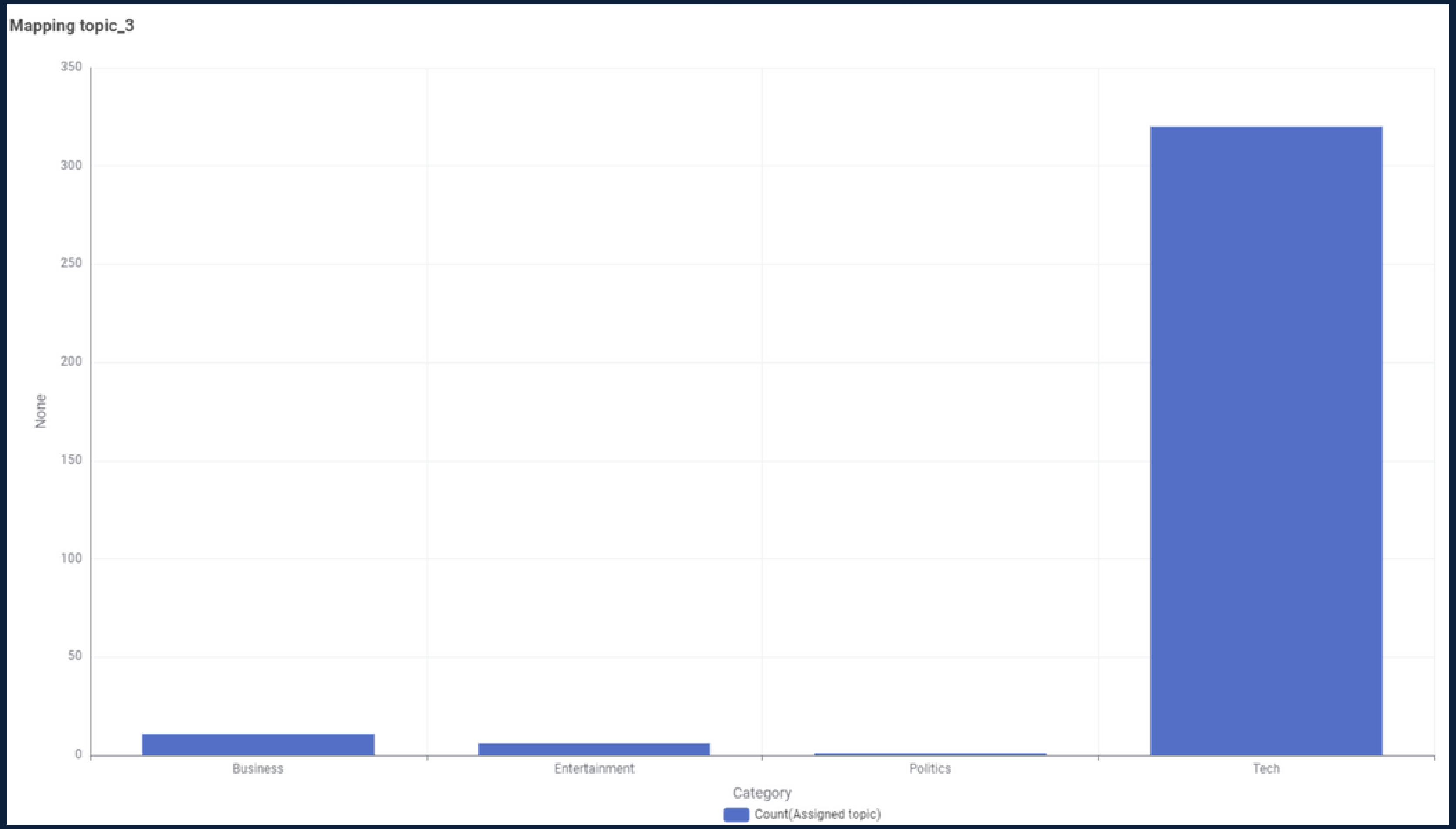
Distribution of Categories for Each Topic ID

topic_3

▶ 1: Group table   Flow Variables

Rows: 4 | Columns: 2   Table   Statistics

#	RowID	Category (String)	Count(Assigned topic) (Number (integer))
1	Row0	Business	11
2	Row1	Entertainment	6
3	Row2	Politics	1
4	Row3	Tech	320



We can say topic_3 is mostly assigned to Tech articles

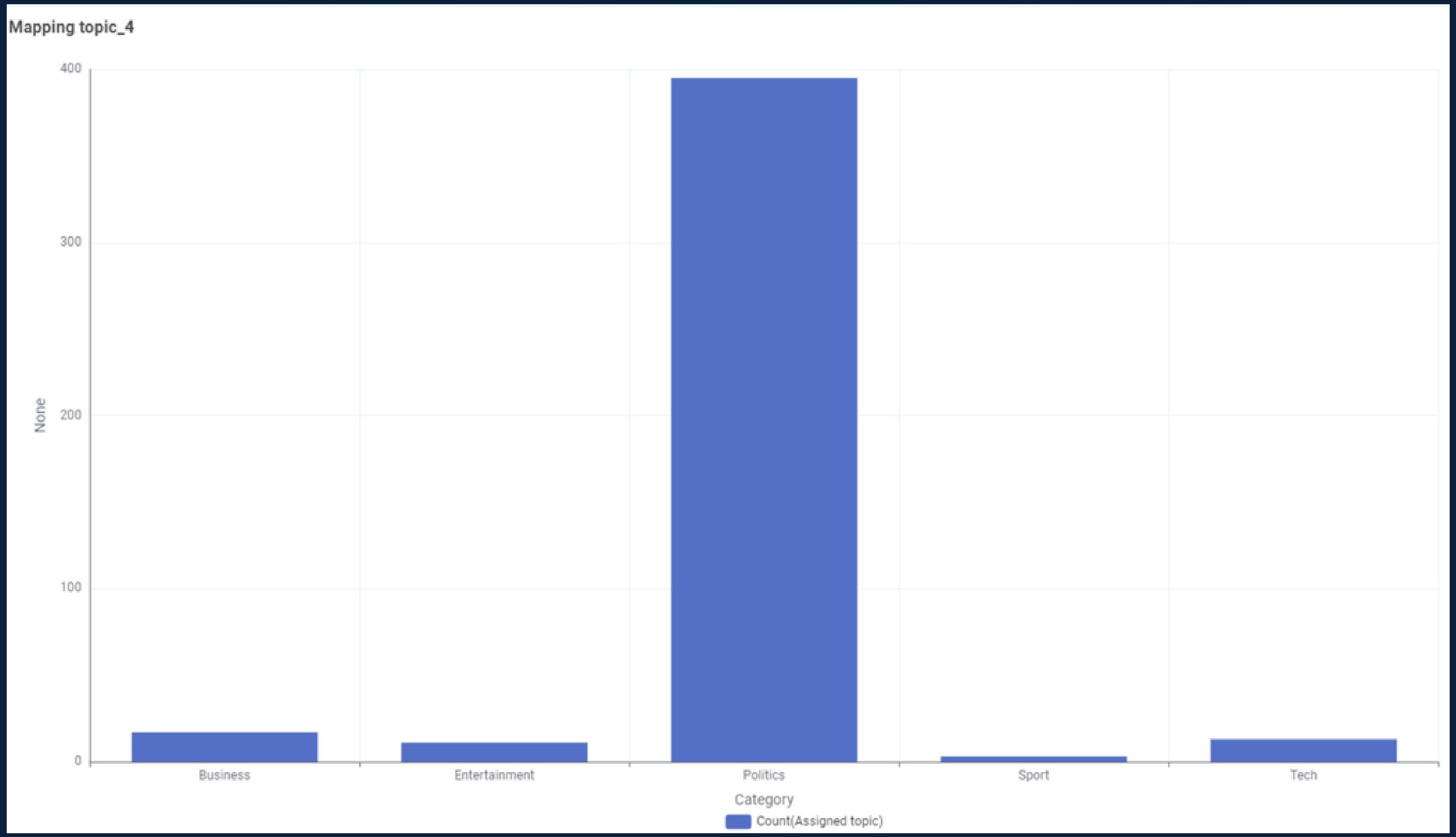
Mapping Topic IDs to Category

Mapping topic_4

1: Group table | Flow Variables

Rows: 5 | Columns: 2 — Table | Statistics

#	RowID	Category (String)	Count(Assigned topic) (Number (integer))
1	Row0	Business	17
2	Row1	Entertainment	11
3	Row2	Politics	395
4	Row3	Sport	3
5	Row4	Tech	13



We can say topic_4 is mostly assigned to Politics articles

LDA

Visualizations - Word Cloud For Ech Topic ID

topic_0

topic_1

topic_2

topic_3

topic_4

Overall

Model Performance Comparison

Topic Modeling

Model	Accuracy	Precision	Recall	F1 Score
LDA Topic Modeling with Count Vectorization	0.911	0.921	0.904	0.910
LSA with TFIDF Vectorization	0.160	0.032	0.200	0.055
BTM with Count Vectorization	0.946	0.945	0.945	0.944
GSDMM (Dirichlet Mixture)	0.168	0.033	0.200	0.056
LDA (Gibbs Sampling)	0.576	0.448	0.549	0.471

Text Clustering

Model	Accuracy	Precision	Recall	F1 Score
KMeans Clustering	0.707	0.642	0.764	0.681
Agglomerative Clustering	0.931	0.932	0.927	0.929
HDBSCAN Clustering	0.470	0.404	0.599	0.408

Model Inferences & Conclusion:

- No single method consistently outperformed the others, reflecting the findings of the health text study.
- BTM and LDA (Grid Search) delivered the strongest topic modeling performance, closely aligning with true categories.
- LSA and GSDMM performed poorly, likely due to sparse signal and vocabulary overlap.
- Among clustering methods, Agglomerative Clustering (with BERT) achieved near-top performance, comparable to the best topic models.
- KMeans produced moderate results, while HDBSCAN underperformed on high-dimensional semantic data.
- Semantic clustering can rival traditional topic models on structured, longer text.
- Method selection should depend on task goals (interpretability vs. accuracy) and data type (short vs. long, formal vs. informal).

THANK YOU