DS 203

Reconstructing Lecture Summaries using NLP Techniques

Presented by
Param Mehta -23B2439
Harsh Chopra - 23B2407
Preksha PC -23B2413
Mohit Doke - 23B2443

OBJECTIVE

Conduct EDA on session and summary data.
Reconnect sessions with summaries using text embeddings.
Visualize overlaps between sessions and keyword distributions.
Rank summaries by relevance and detail.
Build a simple app to retrieve top summaries based on user-input keywords.

EXECUETIVE OVERVIEW

1. Project Objectives
2. Failed attempts
3. TF_IDF Aproach

- EDA
- HDBScan
- PCA vs UMAP
- Heat Map
- Graph based session overlap both

EXECUETIVE OVERVIEW

4 SENTENCE BERT approach
4.1 HDBScan clustering
4.2 UMAP
4.3 Heat map
4.4 Graph based session semantic analysis
4.5 Ranking
4.6 Bubble chart
4.7 Word cloud before and after removing stop words
4.8 Application
4.9 Conclusion
4.10 Evaluation

TF – IDF + HDBSCAN

DATASET

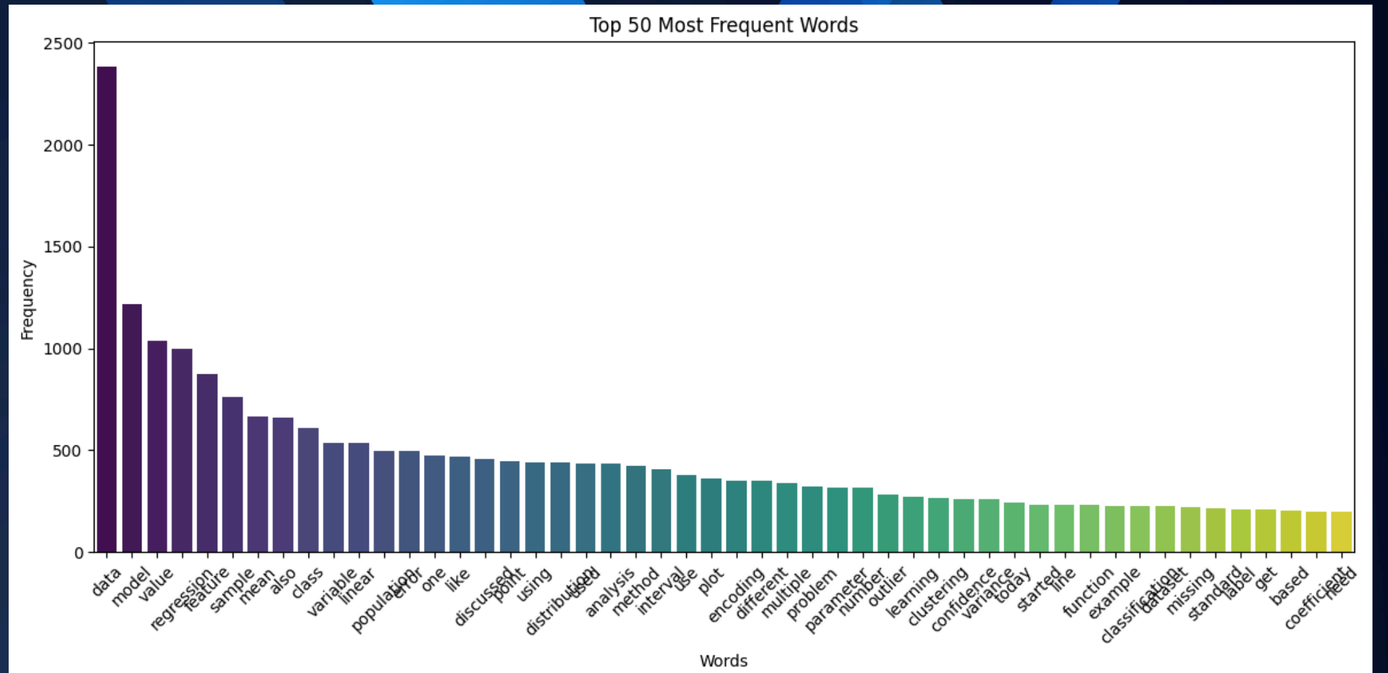
- Dataset contains 668 entries of session summaries in random order, not grouped by sessions
- Cleaned the Session_Summary column by removing links, empty, and duplicate summaries.
- Applied tokenization, stopword removal, and lemmatization to create meaningful tokens.
- Built a vocabulary of unique tokens for further vectorization and similarity analysis.

EXPLORATORY DATA ANALYSIS (EDA)

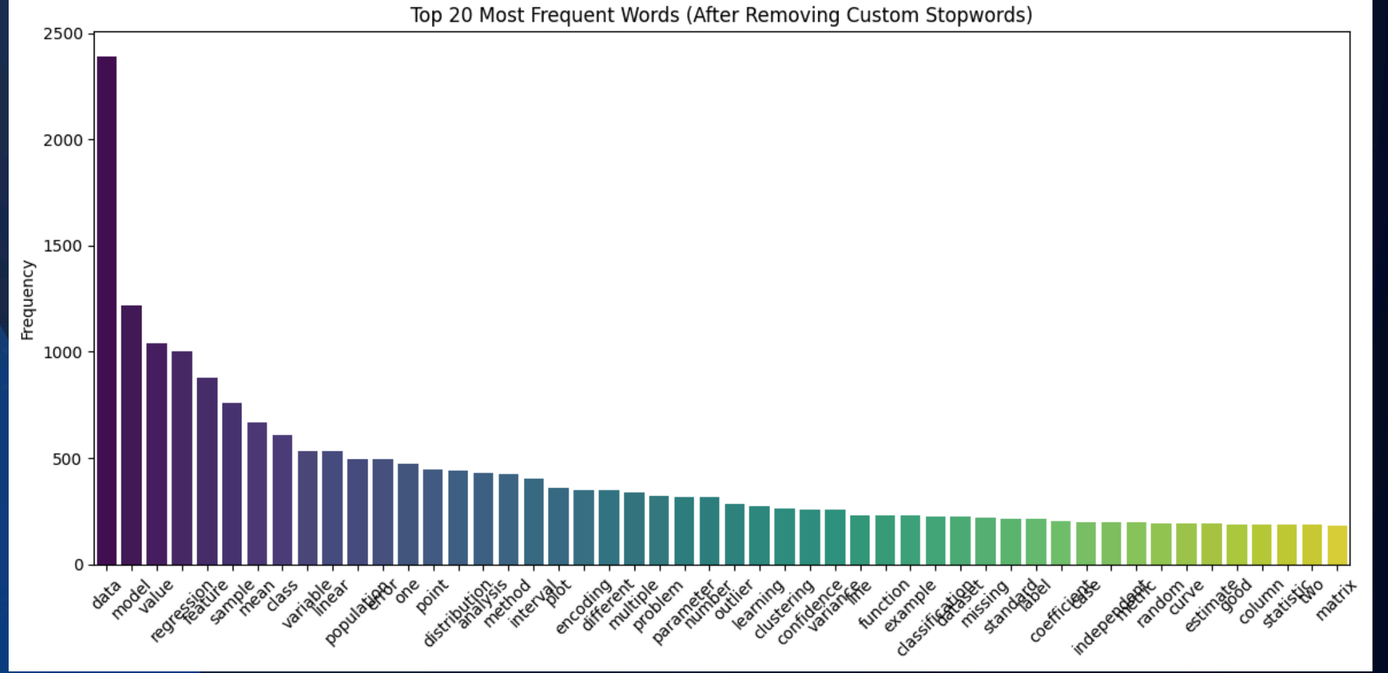
- Because TF-IDF requires data after removing stopwords as it works on Frequency of rare words rather than semantic analysis
- Used frequency distribution to understand common topics/themes
- Some words like "model", "value" appear frequently — indicates common lecture topics, can be dropped
- Token count per summary shows variation in detail level by students

Defined custom stopwords to remove generic, frequent but non-informative words (e.g., data, value, example, used) and regenerated cleaned_summary using updated tokens,
Exactly which words should be droped was determined by trail and error

BEFORE



AFTER



FAILED ATTEMPTS

Doc2vec

Through this we are getting 14 clusters but silhouette score is coming very less .

```
--- Quantitative Evaluation ---
Silhouette Score: 0.0653  (↑ higher is better, range -1 to 1)
Davies-Bouldin Index: 3.6020  (↓ lower is better)
```

FAILED ATTEMPTS

We also tried normalising the data but it oversimpilified the data and it gave only 9 clusters with high silhouette score and considered alot of noise points.

```
Cluster Distribution:
-1    146
 0     32
 1     41
 2    102
 3     94
 4     54
 5     30
 6     46
 7     72
 8     49
Name: count, dtype: int64
```

HDBSCAN

We tried K-means clustering but it was giving bad results so we explored other clustering algorithms and we discovered that for Text data HDBScan is better as it works well in case of data with high dimensionality. It is a density based algorithm

VECTORIZATION METHODS

- Applied TF-IDF Vectorization with: Unigrams + Bigrams (Top 1000 features)
- Dimensionality reduced using Truncated SVD (50 components)
- We do the clustering using cosine similiarity because with Euclidean distance, even small differences in sparse vectors can lead to large distances while cosine uses the angle
- Used HDBSCAN for density-based clustering

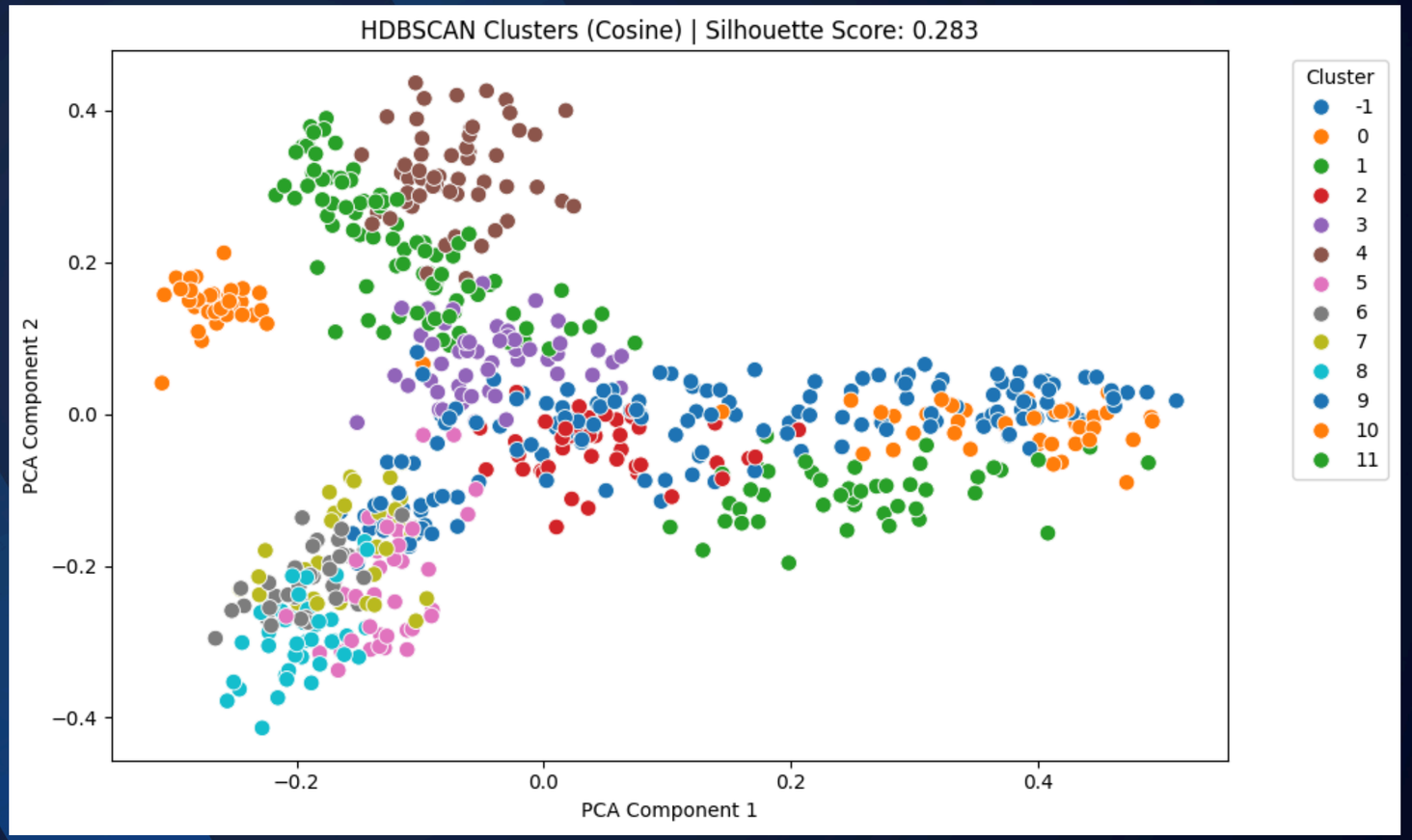
- Visualized clusters using PCA (2D projection)
- Cluster quality metrics (excluding noise points):
  - Silhouette Score: 0.283
  - DBCV Score: 0.135
- The silhouette score indicates that the clusters are weak and the DBCV which measures the clustering wrt density also indicates weak clustering
- Labels reveal lecture/topic-wise grouping trends

CLUSTERING

```
Cluster Distribution:
--------------------------------
Number of clusters: 12
Number of noise points: 109

Elements per cluster:
Cluster 1: 88 elements
Cluster 4: 55 elements
Cluster 3: 52 elements
Cluster 9: 49 elements
Cluster 11: 44 elements
Cluster 2: 43 elements
Cluster 10: 42 elements
Cluster 5: 41 elements
Cluster 8: 41 elements
Cluster 6: 38 elements
Cluster 0: 34 elements
Cluster 7: 30 elements
```

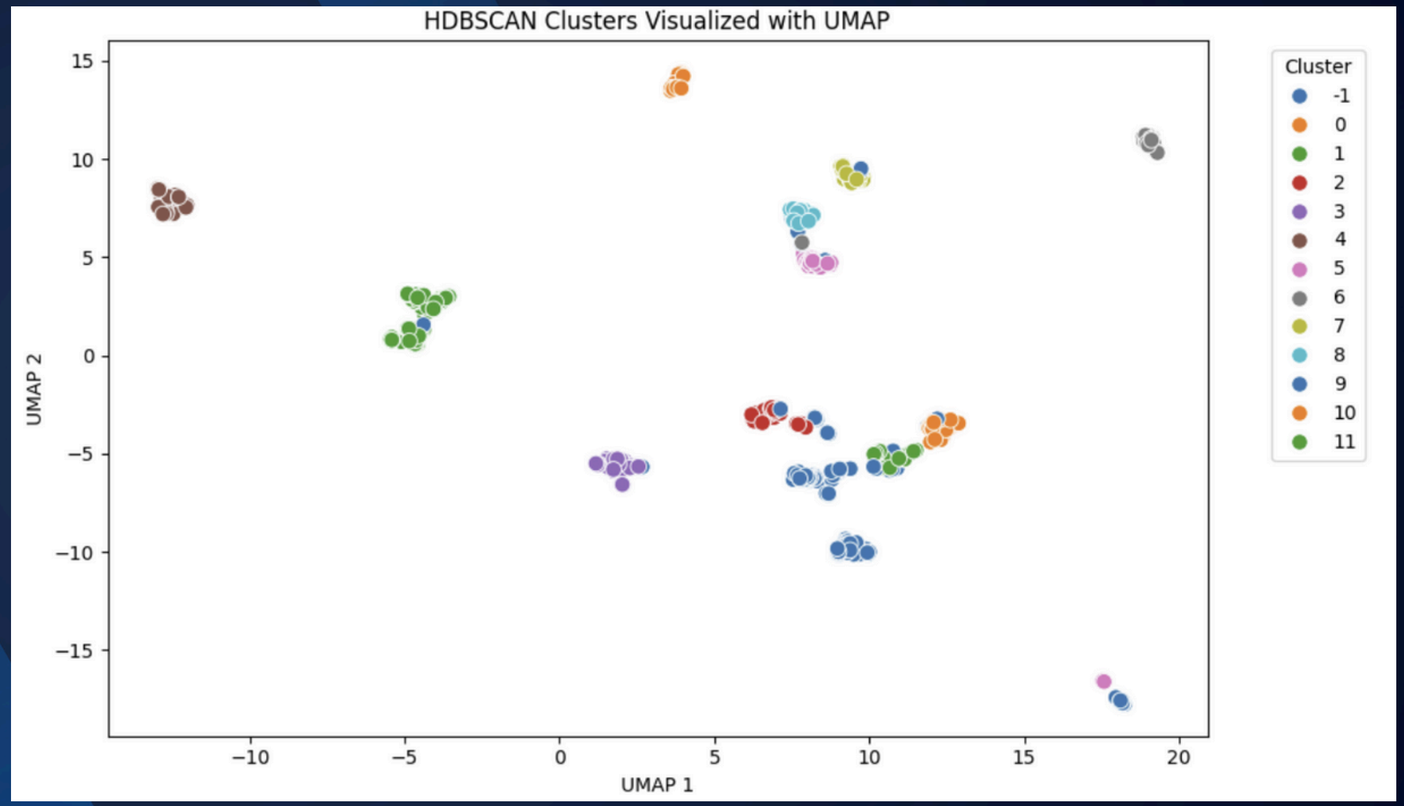
We tried counting the clusters manually it was around 14-15 so 11 seemed like a decent estimate of the clusters. The number of sessions in each cluster were also somewhat even

PCA



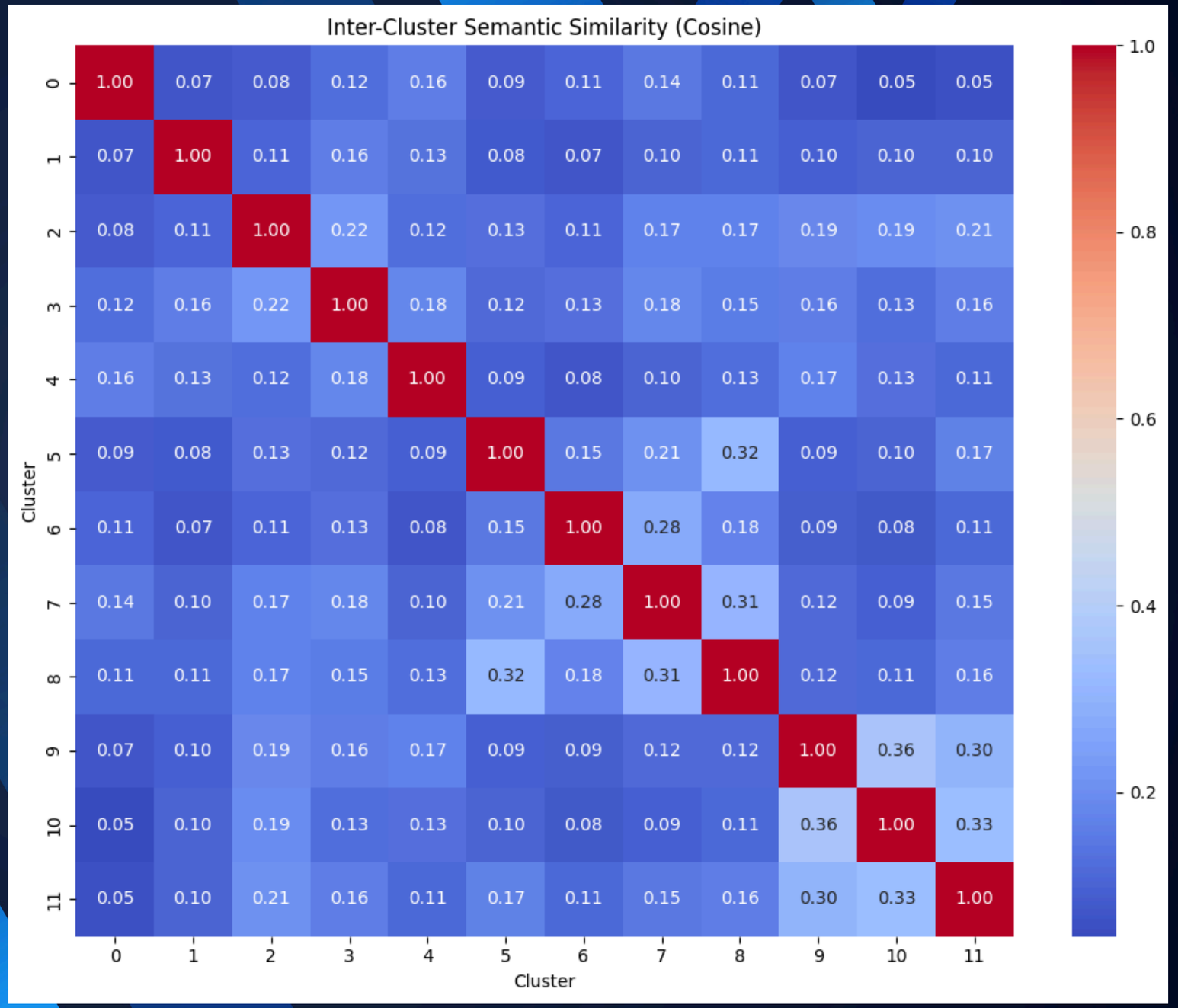
UMAP CLUSTERING

- Applied UMAP on SVD-reduced TF-IDF vectors
- 2D layout clearly shows dense groupings and separation
- Noise points scatter at the edges or between clusters
- UMAP offers better separation of clusters than PCA, useful for interpretation
- The data had high dimensionality hence 2d representations cannot do justice



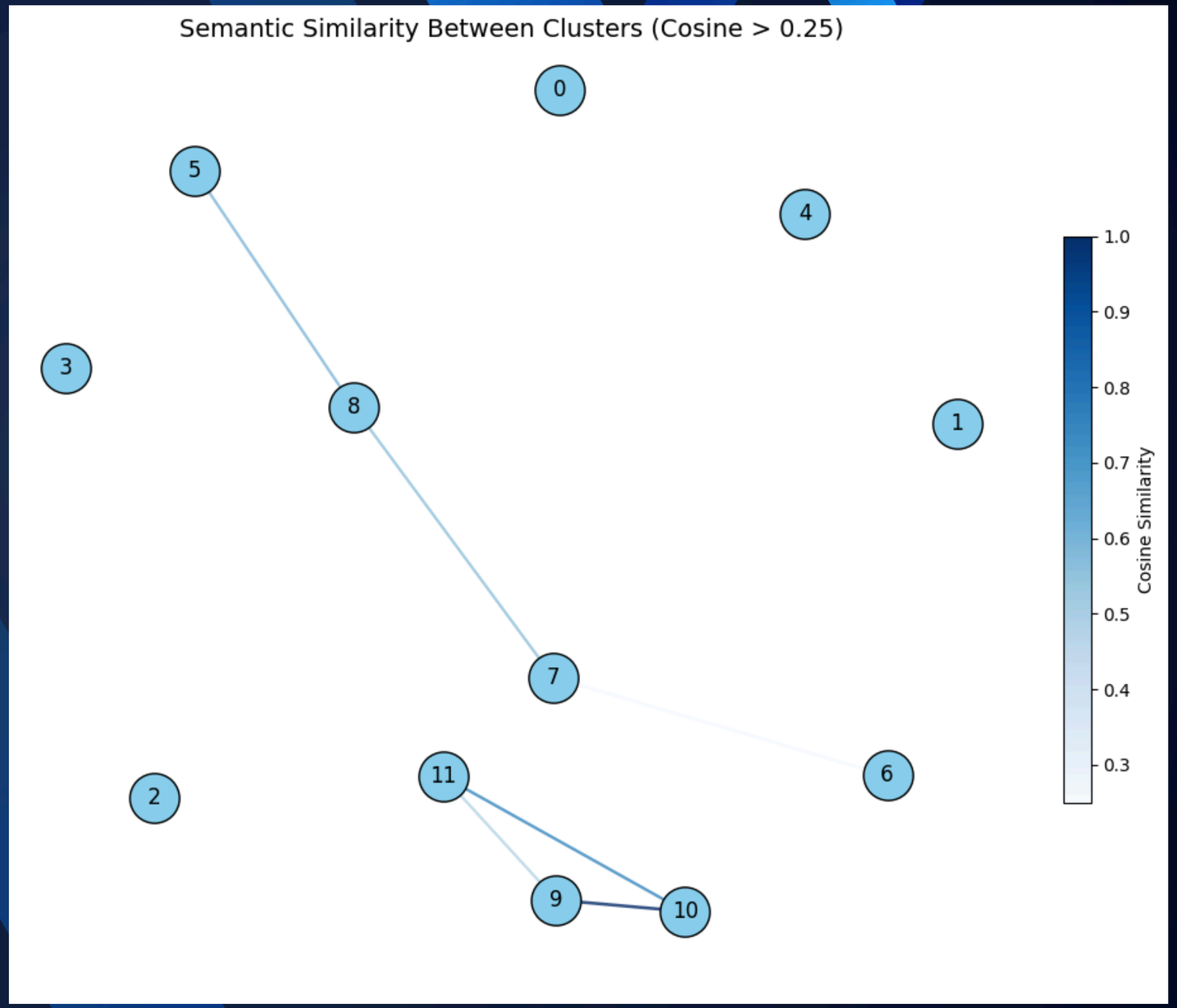
KEY OBSERVATIONS

- Most clusters show low similarity (< 0.2) → indicates clear separation
- Clusters 9, 10, and 11 show higher similarity (up to 0.36) → possible thematic overlap.
- Clusters 5 and 8 also have moderate similarity (0.32)
- Clusters 0, 1, and 5 are well-separated from the rest



SEMANTIC OVERLAP NETWORK

- Using the cluster-wise cosine similarity matrix, built a network where:
  - Nodes = clusters
  - Edges = cluster semantic similarity with a threshold (0.25)
- Edge weights, color + thickness encode similarity (darker = more similar)
- Colorbar shows exact mapping of edge strength.
- This shows triangular similiariyt

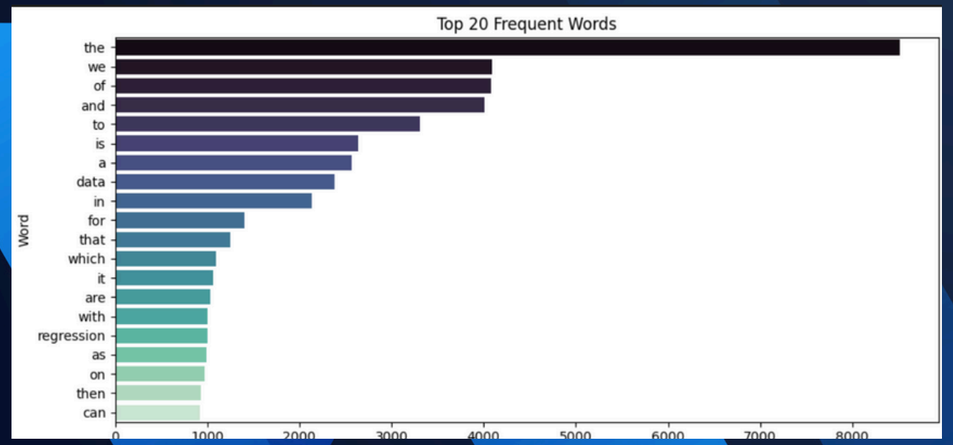


SENTENCE BERT+ HDBSCAN

EXPLORATORY DATA ANALYSIS (EDA)

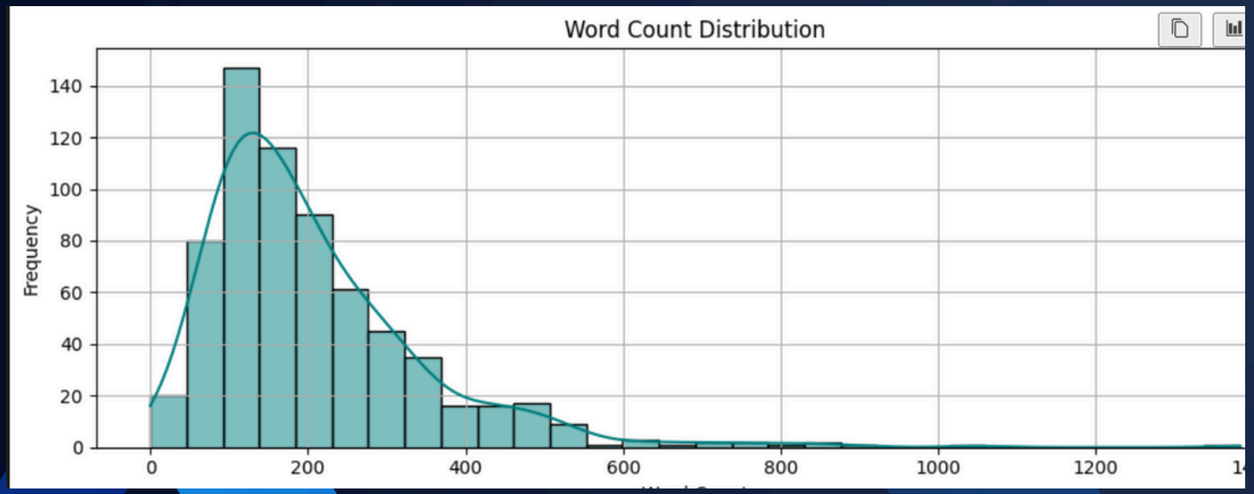
- Used the same df[”cleaned_summary”] as used previously .we get :

```
Number of summaries: 667
Avg Word Count: 208.38
Avg Char Count: 1264.06
```



- The EDA is different for TF-IDF and sentence bert as sentence bert works on the basis of an LLM which does sematic analysis of the data so we do not need to remove the stop words.

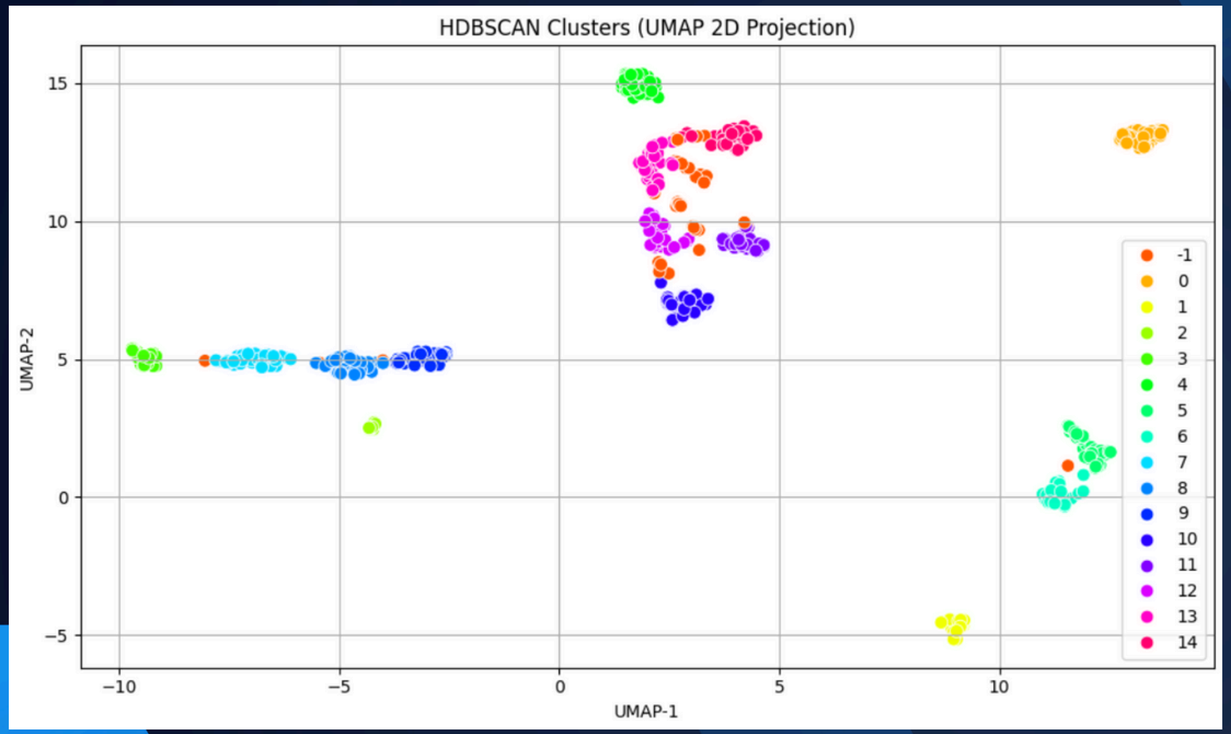
EXPLORATORY DATA ANALYSIS (EDA)



Word Count Distribution

The majority of session texts are between 100 and 250 words, peaking near 150.
The distribution is right-skewed, with a long tail extending beyond 1000 words.
There are a few very long summaries (above 800–1400 words), indicating potential outliers.
The histogram follows a log-normal-like pattern, common in natural language word count distributions.

Here we have got 14 clusters , silhouette score of 0.7934 and the noise points came to be 46 . The estimate was 15 clusters so 14 is a good result

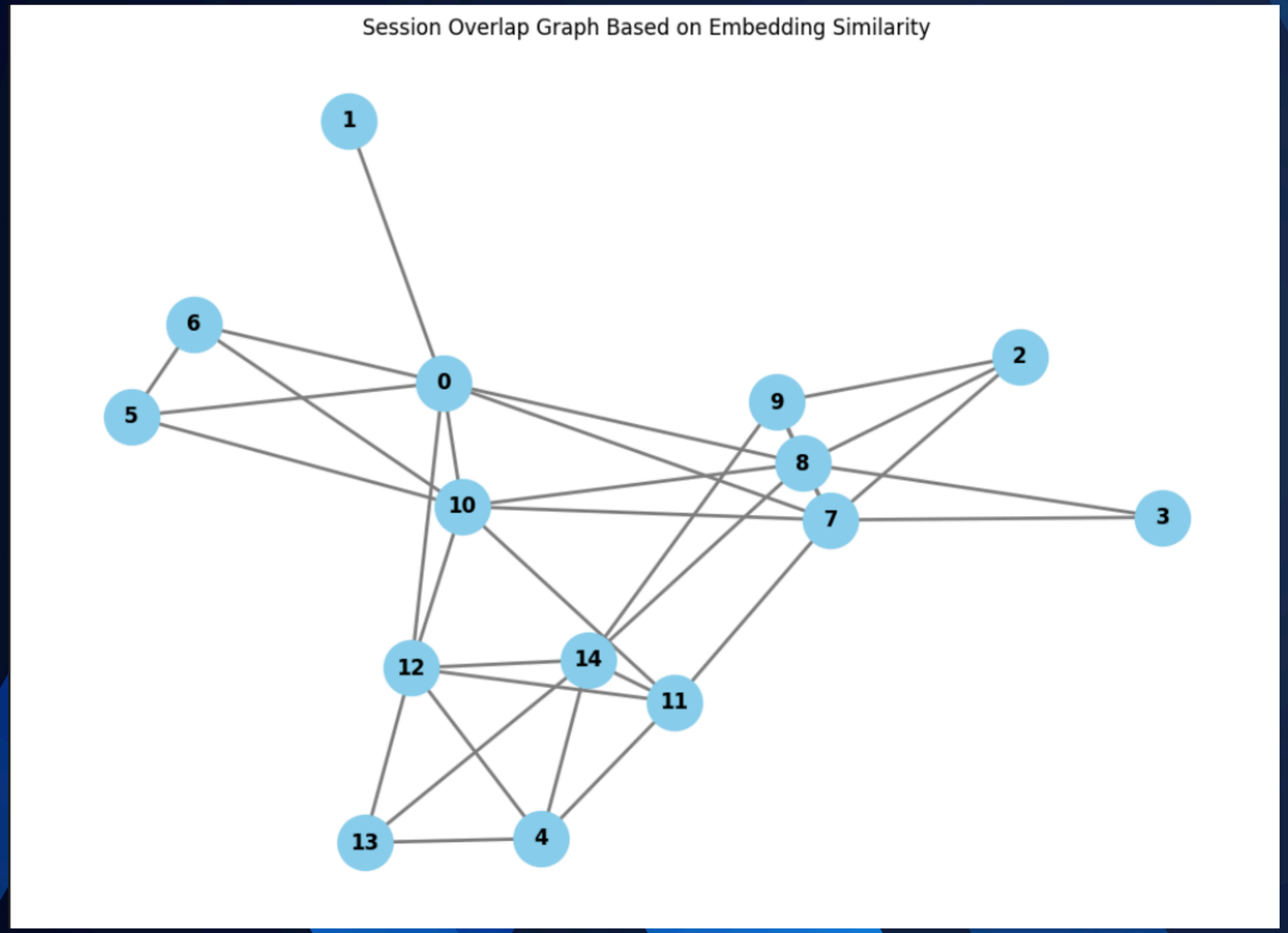



```
🔹 Number of Elements in Each Cluster:
Noise (-1): 43 summaries
Cluster 0: 56 summaries
Cluster 1: 34 summaries
Cluster 2: 14 summaries
Cluster 3: 36 summaries
Cluster 4: 53 summaries
Cluster 5: 49 summaries
Cluster 6: 39 summaries
Cluster 7: 39 summaries
Cluster 8: 45 summaries
Cluster 9: 45 summaries
Cluster 10: 52 summaries
Cluster 11: 34 summaries
Cluster 12: 34 summaries
Cluster 13: 45 summaries
Cluster 14: 48 summaries
```

```
DBCV Score: 0.5926 (↑ higher is better, range -1 to 1)
```

HDBSCAN Clusters: The UMAP plot reveals distinct clustering with labels ranging from 0 to 14, while -1 denotes outliers. Clusters such as 3, 6, and 9 are notably compact, indicating strong semantic cohesion. The dispersed -1 points suggest ambiguous or noisy session summaries that don't align with dominant themes

Graph Relation



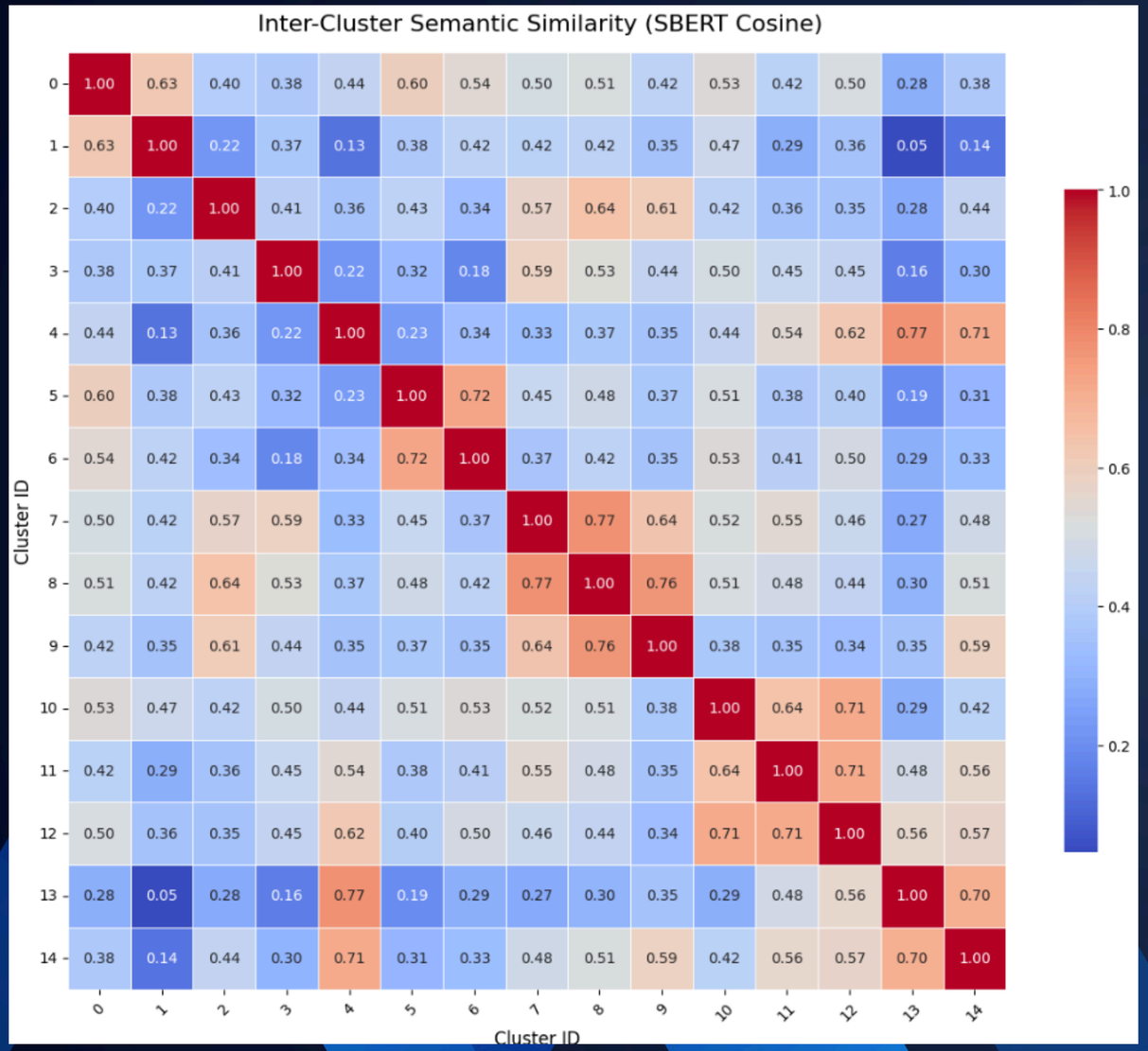
High Centrality Nodes:
Nodes 0, 6, 10, and 13 have many connections, acting as semantic hubs across sessions.

Dense Subgraph:
There's a dense cluster formed by nodes 6, 7, 8, 9, 10, 11, and 13, indicating strong thematic overlap.

Peripheral Nodes:
Nodes like 1 and 4 have fewer connections, implying more unique or isolated content.

Collaborative Structure:
The graph suggests significant session interlinkage, pointing to shared themes across multiple clusters.

SBERT Cosine



The heatmap shows greater similiarity compared to that after removing stopwords as the stopwords are quite common

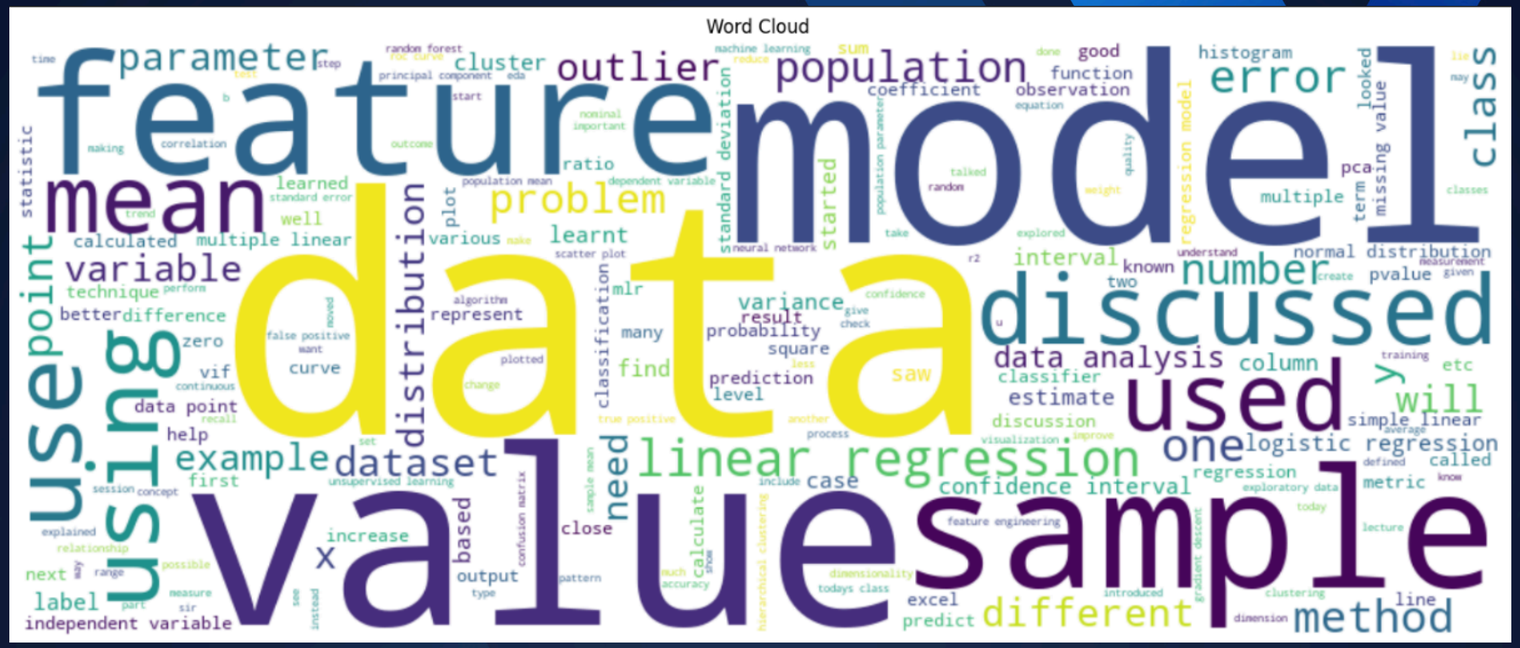
Strongest Connections:
Cluster pairs (6,8), (8,10), and (5,13) show high cosine similarity (~0.76+), indicating strong thematic overlap.

Low Similarity Outliers:
Cluster 1 shows weak similarity with most others, suggesting it contains more distinct content.

Dense Middle Block:
Clusters 6 to 11 share moderate-to-high similarity, forming a semantically cohesive group.

Diagonal Dominance:
All diagonals are 1.0, confirming perfect self-similarity as expected in such matrices.

Word Cloud



The most prominent terms like "sample," "data," "model," "feature," and "value" suggest a strong focus on data science fundamentals and machine learning.

Words such as "regression," "classification," "cluster," and "error" indicate frequent discussion of supervised and unsupervised learning techniques.

Statistical terms like "mean," "interval," "population," "estimate," and "confidence" reflect a substantial emphasis on inferential statistics and analysis.

Usage of verbs like "discussed," "used," "learnt," and "started" highlights a lecture or session-based educational context, where concepts were actively explained or explored.

RANKING OF THE SUMMARIES

Ranking the summaries and returning cleaned summaries

```
for i in range(len(df_filtered)):
    cluster_id = df_filtered.loc[i, 'Cluster']
    cluster_indices = df_filtered[df_filtered['Cluster'] == cluster_id].index.tolist()
    cluster_vectors = tfidf_normed[cluster_indices]
    centroid = np.asarray(cluster_vectors.mean(axis=0))
    sim = cosine_similarity(tfidf_normed[i], centroid)[0][0]
    relevance_scores.append(sim)

df_filtered['cluster_relevance'] = relevance_scores

# Step 7: Composite Score (weights can be adjusted)
df_filtered['composite_score'] = (
    0.5 * df_filtered['cluster_relevance'] +
    0.3 * (df_filtered['tfidf_norm'] / df_filtered['tfidf_norm'].max()) +
    0.2 * (df_filtered['word_count'] / df_filtered['word_count'].max())
)

# Step 8: Ranking within cluster
df_filtered['cluster_rank'] = df_filtered.groupby('Cluster')['composite_score'].rank(ascending=False)

# Step 9: Display top 5 per cluster
ranked_summaries = df_filtered.sort_values(['Cluster', 'cluster_rank'])

for cluster_id in sorted(df_filtered['Cluster'].unique()):
    print(f"\n=== Cluster {cluster_id} ===")
    top = ranked_summaries[ranked_summaries['Cluster'] == cluster_id].head(5)
    for i, row in top.iterrows():
```

```
=== Cluster 0 ===
  Rank 1: we started with the equation yfx for fluid flowing in pipe where we have the values of
  Rank 2: first of all we learnt about names of different types of techniques like slr which is
  Rank 3: so todays discussion start with two types of machine learning model which is supervise
  Rank 4: ds 203 15 january 2025 3rd lecture
sir started by talking about y  fx he first talked about old and new methods of data analysis fo
  Rank 5: in this lecture we started by looking at how results were used to be predicted earlier

=== Cluster 1 ===
  Rank 1: feature encoding techniques

during this session we reviewed several techniques used to transform categorical data into numer
  Rank 2: feature encoding when either the dependent variable or some of the independent variabl
  Rank 3: today we started the discussion with feature encoding methods when either of the deper
  Rank 4: in this session we explored various feature encoding techniques essential for converti
  Rank 5: in todays class we covered different techniques for encoding categorical data we start

=== Cluster 2 ===
```

Ranking the summaries and returning original summaries

```
# Step 9: Attach original summaries for display

df_filtered = pd.merge(
    df_filtered,
    df[['cleaned_summary', 'Session_Summary']],
    on='cleaned_summary',
    how='left'
)

# Step 10: Display top 5 per cluster using original summary text
ranked_summaries = df_filtered.sort_values(['Cluster', 'cluster_rank'])

for cluster_id in sorted(df_filtered['Cluster'].unique()):
    print(f"\n=== Cluster {cluster_id} ===")
    top = ranked_summaries[ranked_summaries['Cluster'] == cluster_id].head(5)
    for i, row in top.iterrows():
        print(f"  Rank {int(row['cluster_rank'])}: {row['Session_Summary'][:150]
```

```
=== Cluster 0 ===
  Rank 1: we started with the equation y=f(x) (for fluid flowing in pipe) wh
  Rank 2: first of all we learnt about names of different types of technique
  Rank 3: so today's discussion start with two types of machine learning moc
  Rank 4: ds 203 15 january, 2025 (3rd lecture)
sir started by talking about y = f(x). he first talked about old and new met
  Rank 5: in this lecture we started by looking at how results were used to

=== Cluster 1 ===
  Rank 1: feature encoding techniques

during this session, we reviewed several techniques used to transform catego
  Rank 2: feature encoding. when either the dependent variable or some of th
  Rank 3: today we started the discussion with feature encoding methods. whe
  Rank 4: in this session, we explored various feature encoding techniques,
  Rank 5: in todayâ€™s class, we covered different techniques for encoding c
```

Bubble Chart

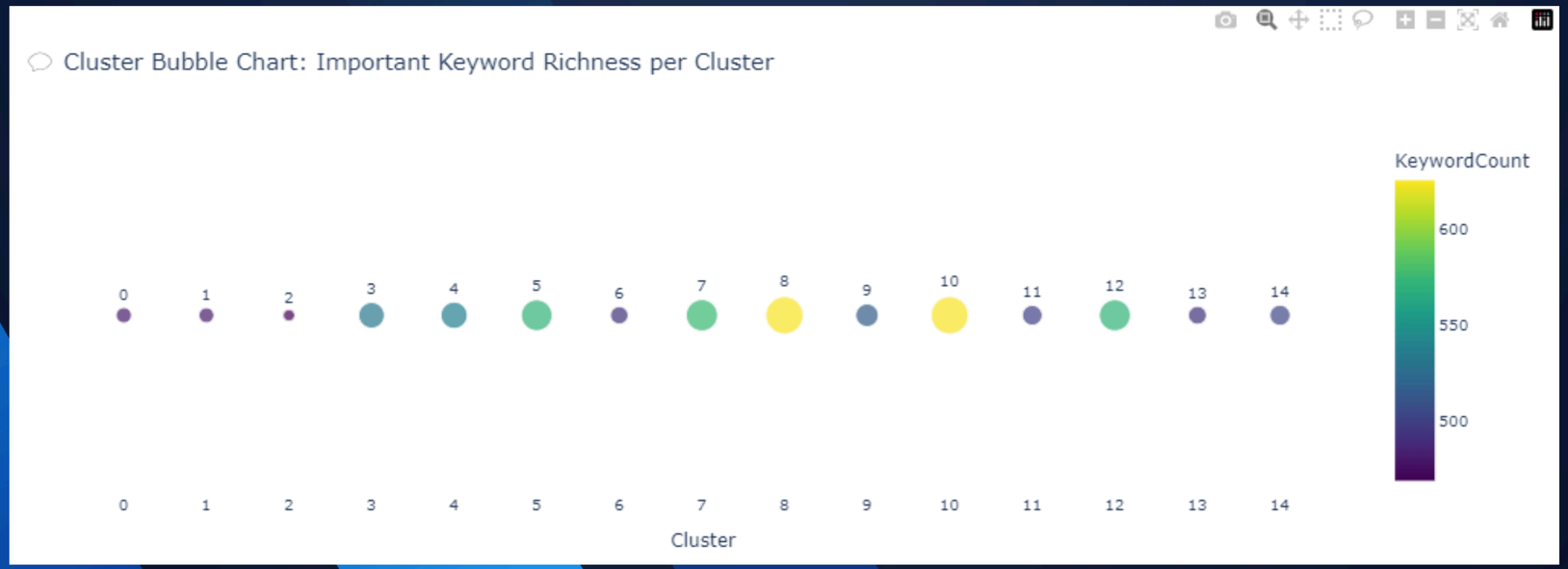
```
# Recompute combined summary in a separate step since we reused the column
cluster_summary["combined_summary"] = df_filtered.groupby("Cluster")["cleaned_summary"].apply(lambda x: " ".join(x))

# --- TF-IDF to extract keyword importance ---
tfidf = TfidfVectorizer(max_features=1000, stop_words='english')
tfidf_matrix = tfidf.fit_transform(cluster_summary["combined_summary"])
important_keyword_counts = (tfidf_matrix > 0).sum(axis=1).A1  # Count of unique important keywords per cluster

# --- Normalize bubble sizes ---
scaler = MinMaxScaler(feature_range=(10, 100))
bubble_sizes = scaler.fit_transform(important_keyword_counts.reshape(-1, 1)).flatten()

# --- Build DataFrame for Plotly ---
bubble_df = cluster_summary.copy()
bubble_df["KeywordCount"] = important_keyword_counts
bubble_df["BubbleSize"] = bubble_sizes
bubble_df["Cluster"] = bubble_df.index.astype(str)

# --- Plot ---
fig = px.scatter(
    bubble_df,
    x="Cluster",
    y=[1] * len(bubble_df),
    size="BubbleSize",
    color="KeywordCount",
    color_continuous_scale="Viridis",
    text="Cluster",
```

Cluster 3 stands out with the largest bubble, indicating it has the highest keyword richness (over 650 keywords).

Clusters 7 and 9 also show higher keyword counts compared to others, though their bubbles are smaller than Cluster 3.

Clusters 0, 1, 2, 11, 12, and 13 have smaller bubbles, suggesting lower keyword richness (closer to 500 keywords).

The color gradient shows that keyword richness generally increases from left to right across clusters, with a few exceptions like Cluster 6, which is notably rich.



Getting the top summaries

We save the top summaries in “top_summaries_per_cluster.csv

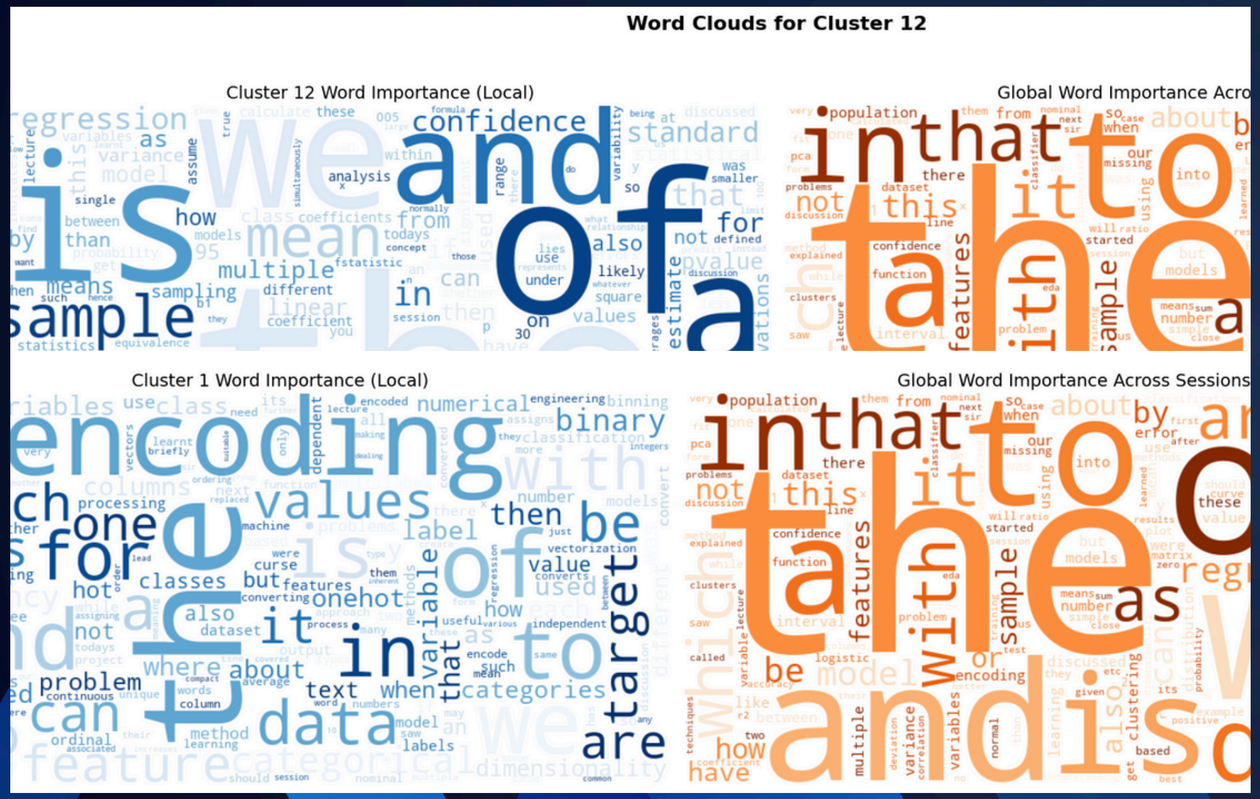
```
# Save top 5 summaries of each session from the original summaries per cluster to CSV
top_summaries = ranked_summaries.groupby('Cluster').head(5)

# Use the original (uncleaned) session summaries column after merge
top_summaries[['Cluster', 'cluster_rank', 'composite_score', 'Session_Summary_original']].to_csv('top_summaries_per_cluster.csv', index=False)
```

NOTE - The summaries would be displayed through the application and these summaries are original i.e. they are not cleaned.

WORD CLOUD

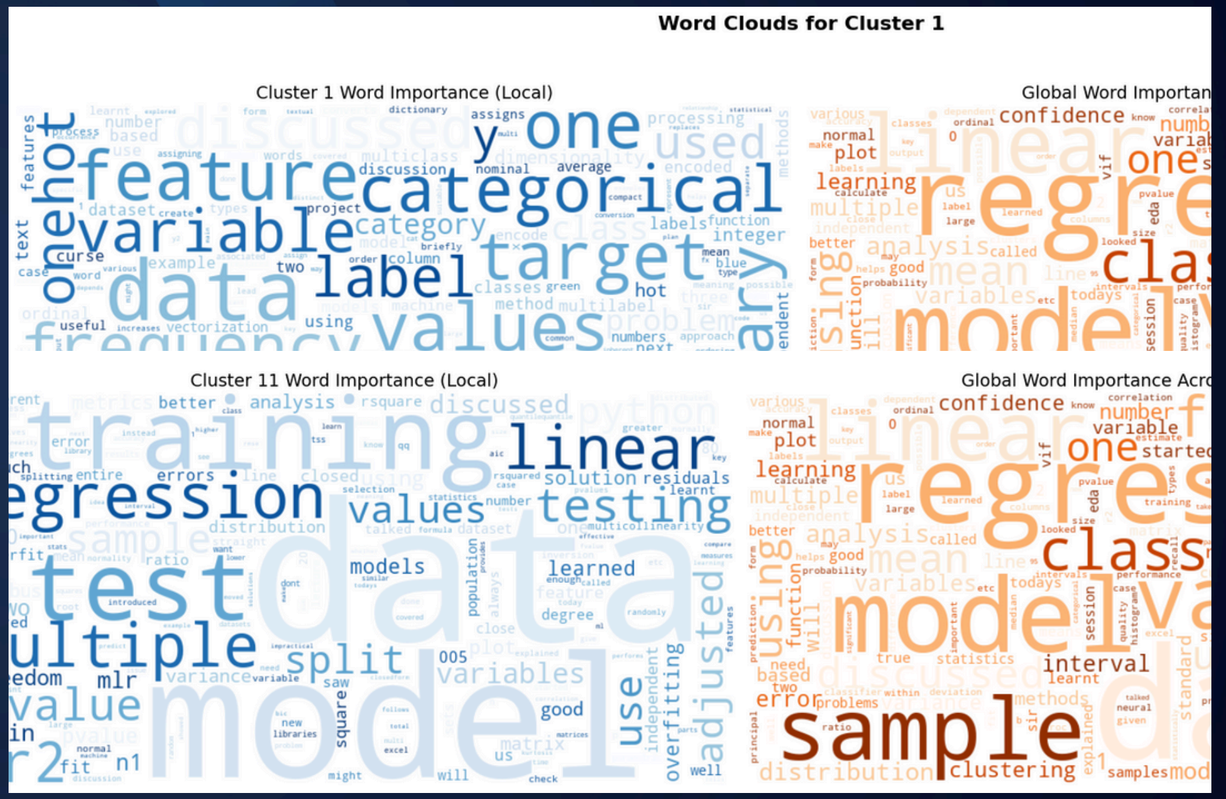
As we see that without stopwords the majority of the words are “is” , “of” etc which are not important in the summary

Without using stopwords



For all the clusters

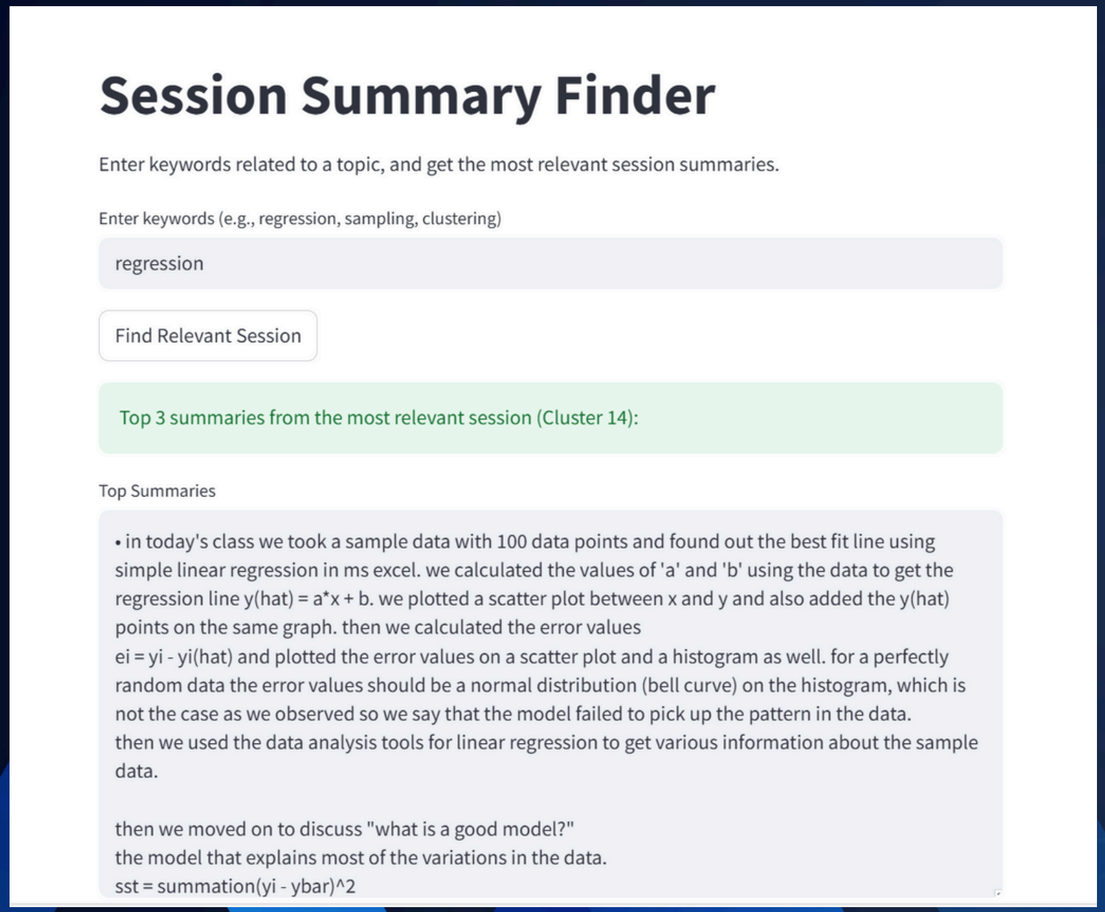
With use of stopwords



For all the clusters

Final Application

Final project github link




```python
# Load data
df = pd.read_csv("top_summaries_per_cluster.csv")

# Load Sentence-BERT model
model = SentenceTransformer('all-MiniLM-L6-v2')

st.title("Session Summary Finder")
st.write("Enter keywords related to a topic, and get the most relevant session summaries.")

# Input box for keywords
keywords = st.text_input("Enter keywords (e.g., regression, sampling, clustering)", "")

if st.button("Find Relevant Session"):
    if keywords.strip() == "":
        st.warning("Please enter at least one keyword.")
    else:
        # Encode user query
        query_embedding = model.encode([keywords])

        # Encode summaries from CSV (only once for top summaries)
        df['embedding'] = df['Session_Summary_original'].apply(lambda x: model.encode(x))

        # Compute cosine similarity
        df['similarity'] = df['embedding'].apply(lambda x: cosine_similarity([x], query_embedding)[0][0])

        # Find the session (cluster) with highest average similarity
        best_cluster = df.groupby('Cluster')['similarity'].mean().idxmax()
        top_matches = df[df['Cluster'] == best_cluster].sort_values(by='cluster_rank').head(3)

        st.success(f"Top 3 summaries from the most relevant session (Cluster {best_cluster}):")

        # Show summaries in a text area
        summary_text = "\n\n".join(f"• {row['Session_Summary_original']}" for _, row in top_matches.iterrows())
        st.text_area("Top Summaries", value=summary_text, height=300)
```

CONCLUSION AND LEARNINGS

- Learnt that about various methods that can be used for text featurisation.
- Learnt about differences between K-means and HDBScan for clustering
- Learnt about various scores for determining how good is clustering.

EVALUATION

- How good is the EDA and its documentation?
- We conducted proper EDA for each text featurization method according to its requirements and documented the steps properly

EVALUATION

- Has there been any creative thinking and innovation while solving the problems?
- We also analysed where stop words should be removed and where not, also analysed and found the which method would be best for clustering and visualisation

EVALUATION

- Quality of Feature Engineering / Feature Creation in terms of relevance to the problem
- After multiple attempts we figured out using a semantic model for featurization would be best as it would be able to capture the data well.

EVALUATION

- Steps of Data Science: Have they been correctly applied, and backed up with proper metrics / reasons / explanations?
- We analysed our clustering based on Sillhoutte score and DBCV score and it was giving good results hence we finalized that

EVALUATION

- Completeness and correctness of the results achieved
- Upon manual counting we got the total number of sessions to be around 15 and our model also predicts 15 and the clusters are also mostly even.

THANK YOU