DS 203 Project

Classifying Summaries

22B1506, 22B2415, 22B1527

Project Workflow

Problem Understanding & Discussion

Tool Familiarization

Initial Approach & Preprocessing

Exploratory Data Analysis (EDA)

Baseline Modeling

Semantic Boost with Word2Vec

Tried & Compared Other Featurization Techniques

Visual Exploration with UMAP

Ranking Logic Implementation

Word Clouds & Session Visualization

App Building

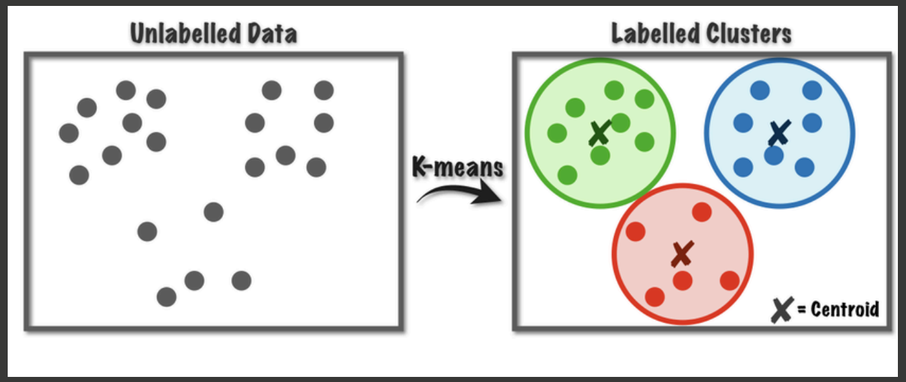
Challenges Faced

Final Conclusion

Unsupervised Learning: K-Means Clustering

Why Unsupervised Learning?
Our dataset lacked labels - summaries were not linked to any sessions. We needed to discover patterns and group summaries based on their semantic similarities without predefined categories. This made unsupervised learning the natural choice to cluster and explore the hidden structure in the data.

A popular unsupervised algorithm that partitions data into K clusters, where each data point belongs to the cluster with the nearest mean. Helps in grouping similar summaries to identify which ones might belong to the same session. It is simple, scalable, and efficient for large text embeddings. Works well when clusters are roughly spherical in vector space (which holds in high-dimensional text data after normalization). Helps in initializing the reassociation of summaries to sessions, acting as a first-pass grouping.



- Si: the set of points in cluster i
- μi: the centroid of cluster i

$$\arg\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

What is Silhouette Score?

A metric used to evaluate the quality of clustering. Measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation)

Values range from -1 to 1:

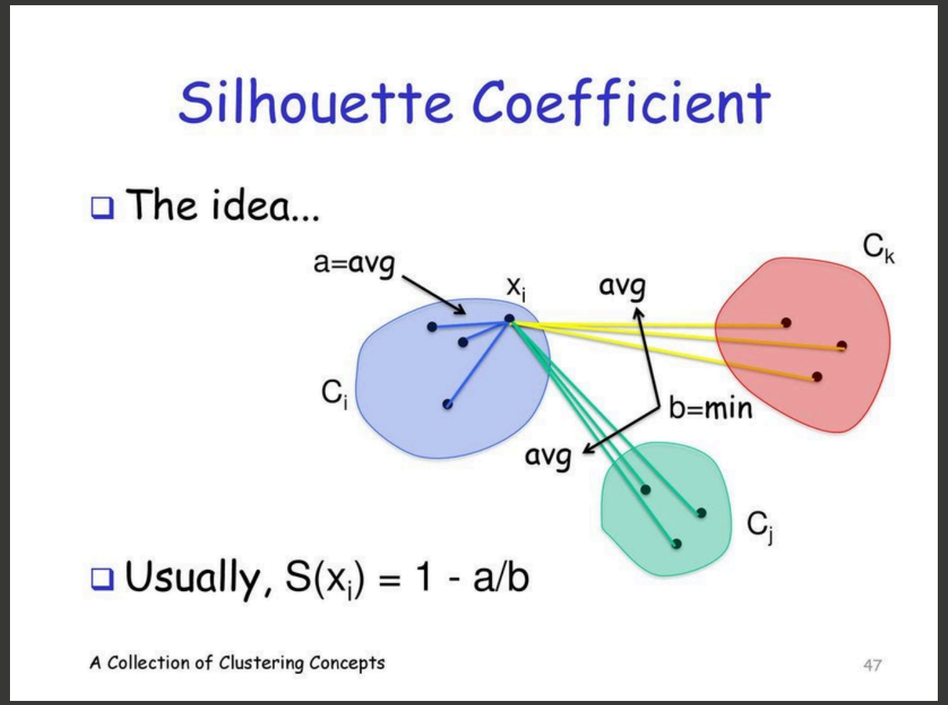
- Close to 1 → well-matched to its cluster, and poorly matched to neighboring clusters
- Around 0 → on the boundary between clusters
- Negative → may be assigned to the wrong cluster

Then, the Silhouette Score for a single point i is: $$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- a(i) = average distance between point i and all other points in the same cluster (intra-cluster distance)
- b(i) = minimum average distance from point i to all points in the nearest different cluster (inter-cluster distance)

Overall Score for the Dataset: $$S = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

Where n = total number of samples



Why is Silhouette Score important?

No Ground Truth Available

- Since the session-summary associations are lost, we do not have labeled data to directly evaluate model accuracy

Need for an Internal Evaluation Metric

- We are comparing clustering performance across multiple models and featurization techniques (e.g., TF-IDF, BERT).
- → A consistent internal metric is required to assess clustering quality.

Silhouette Score Fits Perfectly

- ✅ Measures how well each summary fits within its assigned cluster vs. others
- ✅ Ranges from -1 (poor clustering) to 1 (excellent clustering)
- ✅ Works without labeled data

Supports Model Selection

- Helps us choose the best clustering method + embedding strategy,
- even when session summaries may not fully represent the entire content of the sessions.

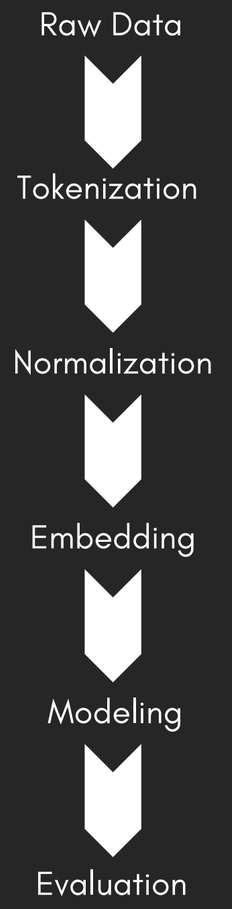
Natural Language Processing

Our project is entirely based on processing and analyzing textual summaries. Hence, a solid understanding of Natural Language Processing (NLP) is essential

Natural Language Processing (NLP) is a subfield of Artificial Intelligence that enables computers to understand, interpret, and generate human language in a meaningful way

Core NLP Principles

- **Text Normalization is Essential**: Always standardize the text (e.g., lowercasing, punctuation and stopword removal) to ensure uniform processing
- **Contextual Understanding is Critical**: Word and sentence meanings change with context. Embedding methods like Word2Vec, BERT, or Sentence Transformers are used to capture this
- **Tokenization is a Mandatory First Step**: Raw text must be split into tokens (words, subwords, or characters) to be processed by NLP models
- **Semantic Similarity Should Be Captured**: Words or summaries may convey the same meaning despite different phrasing. NLP systems should account for this
- **Evaluation Should Match Task Type**: Without labels, use internal evaluation metrics (e.g., Silhouette Score) for tasks like clustering



Text Preprocessing Steps

1. **Word Cleaning (using re library)**
   - Used regular expressions (re module in Python) to:
     - Remove punctuation and special characters
     - Retain only alphabetic characters
     - Normalize whitespace

2. **Tokenization**
   - Splitting the cleaned text into individual words or tokens.
   - Enables further processing like stemming or frequency analysis

3. S**topword Removal**
   - Common words (e.g., “the”, “is”, “and”) that do not contribute to text meaning were removed.

4**. Stemming**
   - Reduces words to their root form by chopping off suffixes (e.g., “running” → “run”)

5. **Lemmatization**
   - Converts words to their base dictionary form (e.g., “better” → “good”)
   - More accurate than stemming.

6. **Lowercasing**
   - Convert all text to lowercase to ensure uniformity

7. **Removing Extra Whitespace**
   - Normalize multiple spaces, tabs, or newlines into a single space using re

8. **Spelling Correction**
   - Fixes typos or misspelled words.
   - Libraries: TextBlob, SymSpell, or pyspellchecker

9. **Named Entity Removal or Tagging**
   - Remove or tag proper nouns like names, places, etc., depending on the task
   - Done using spaCy or nltk's ne_chunk

10. **N-gram Generation**
    - Create bi-grams or tri-grams to preserve contextual information in fe engineering

11. **POS Tagging (for advanced lemmatization or analysis)**
    - Identifies part-of-speech for each word to guide lemmatization.

Text Featurization Methods

Text featurization is the process of converting textual data (words, phrases, sentences) into numerical vectors that can be processed by machine learning models

Why is it Needed?

- Machine learning models work with numerical input, not raw text.
- Featurization captures semantic meaning, context, and syntactic structure of the text.
- It allows us to compare, cluster, or classify text using mathematical operations.

Each word or document is transformed into a point in high-dimensional space based on its usage and context.

- Simple methods (e.g., Bag of Words, TF-IDF) represent word frequency.
- Advanced methods (e.g., Word2Vec, BERT) capture semantic similarity and contextual usage

W**hy High-Dimensional Vectors?**

- Language is complex and multi-dimensional — words have different meanings in different contexts.
- Throwing vocabulary into high-dimensional space helps:
  - Distinguish similar vs. dissimilar meanings
  - Preserve relationships (e.g., king - man + woman ≈ queen)
  - Enable geometric reasoning over text

Bag of Words

What is a Corpus?

A corpus is a collection of text documents used for training or analysis in NLP. In our case, the corpus consists of all the session summaries provided for the project

What is Bag of Words?

Bag of Words is a simple text featurization method where:

- Each document is represented by the frequency of words it contains.
- It ignores word order and grammar.
- The result is a sparse vector where each dimension corresponds to a unique word in the corpus vocabulary

How It Works:

1. Create a vocabulary of all unique words across the corpus.
2. For each document, count how many times each word appears.
3. Represent each document as a vector of word counts.

Document D1	*The child makes the dog happy* the: 2, dog: 1, makes: 1, child: 1, happy: 1
Document D2	*The dog makes the child happy* the: 2, child: 1, makes: 1, dog: 1, happy: 1

↓

	child	dog	happy	makes	the	BoW Vector representations
D1	1	1	1	1	2	[1,1,1,1,2]
D2	1	1	1	1	2	[1,1,1,1,2]

Word2Vec Embedding

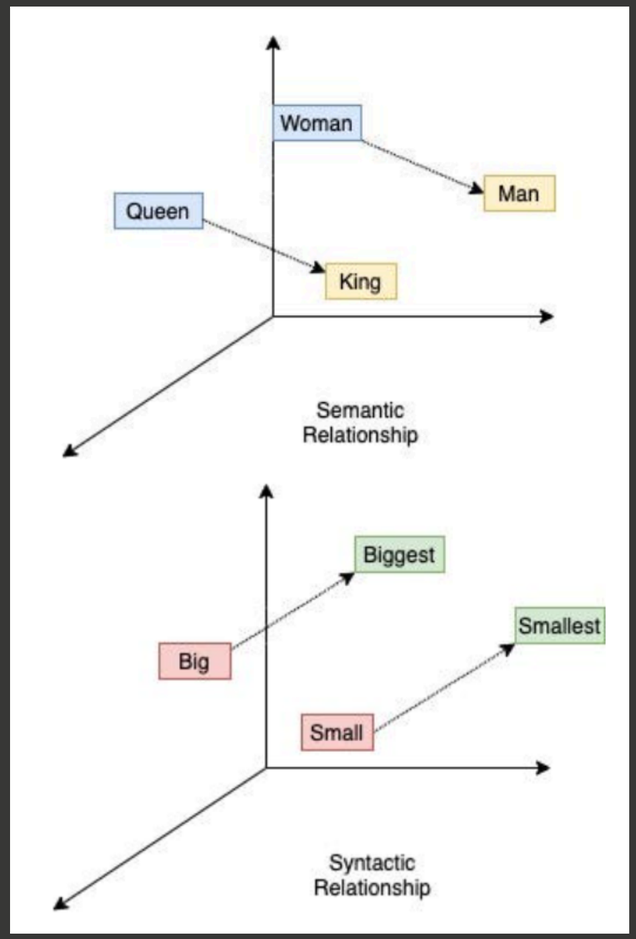
Word2Vec is a predictive, neural network-based technique that converts words into dense, low-dimensional vectors such that semantically similar words are close together in vector space. Unlike Bag of Words, Word2Vec captures context and meaning

Predict surrounding words (context) given a target word (or vice versa)
Two common architectures:

- CBOW (Continuous Bag of Words) - predicts a word from its context.
- Skip-Gram - predicts surrounding context words from a target word.

Words that appear in similar contexts get similar embeddings.

Produces dense vectors (e.g., 100-300 dimensions) instead of sparse ones.
Captures semantic relationships: e.g., king - man + woman ≈ queen
Respects word meaning, not just occurrence.



TF-IDF Scores

TF-IDF (Term Frequency - Inverse Document Frequency) is a statistical method used to reflect how important a word is to a document relative to a corpus.
It improves over simple word counts by downweighting common words and highlighting informative terms

For a word w in document d:

- TF(w, d): Term Frequency → how often word w appears in d
- IDF(w): Inverse Document Frequency

→ log of total number of documents divided by number of documents containing w

$$\text{TF-IDF}(w, d) = \text{TF}(w, d) \times \log\left(\frac{N}{\text{DF}(w)}\right)$$

Why Use TF-IDF?

- Highlights discriminative words in each summary.
- Common words like "the", "and", "of" receive lower scores.
- Useful for document similarity, feature selection, and clustering

$\texttt{tf}(t, d)$

	blue	bright	can	see	shining	sky	sun	today
1	1/2	0	0	0	0	1/2	0	0
2	0	1/3	0	0	0	0	1/3	1/3
3	0	1/3	0	0	0	1/3	1/3	0
4	0	1/6	1/6	1/6	1/6	0	1/3	0

- TF-IDF: Multiply TF and IDF scores, use to rank importance of words within documents
  - Most important word for each document is highlighted

x

$\texttt{idf}(t, D)$

	blue	bright	can	see	shining	sky	sun	today
	0.602	0.125	0.602	0.602	0.602	0.301	0.125	0.602

$\texttt{tfidf}(t, d, D) = \texttt{tf}(t, d) \cdot \texttt{idf}(t, D)$

	blue	bright	can	see	shining	sky	sun	today
1	**0.301**	0	0	0	0	0.151	0	0
2	0	0.0417	0	0	0	0	0.0417	**0.201**
3	0	0.0417	0	0	0	**0.100**	0.0417	0
4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

What is our Problem Statement?

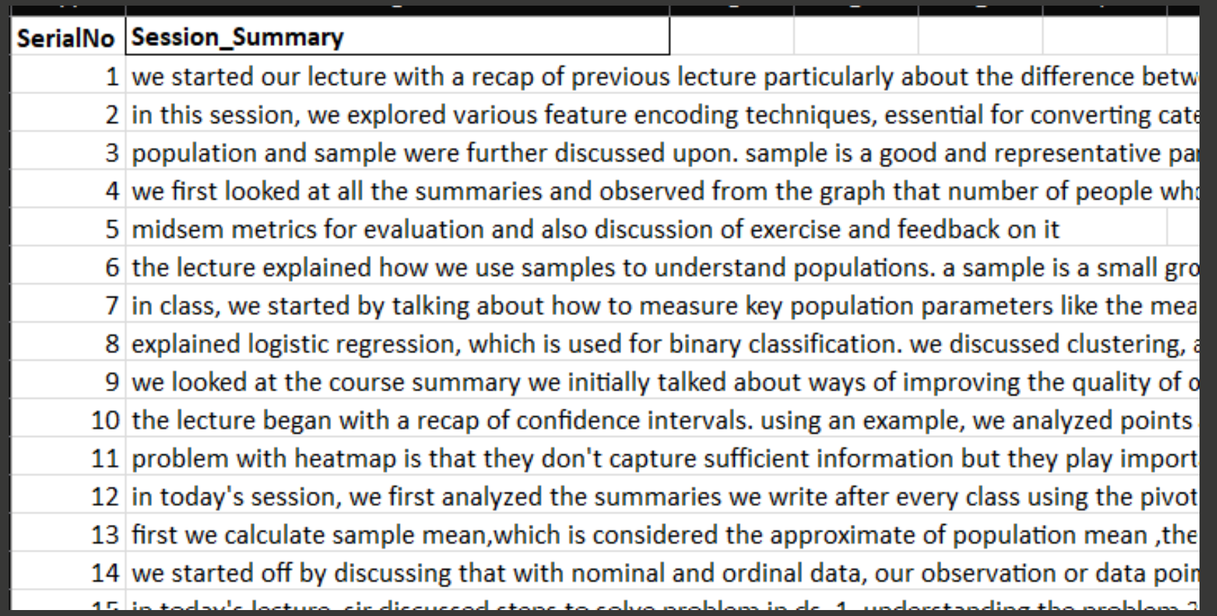
In this project, we are given a dataset of unlabeled session summaries from a Data Science course. Due to poor organization, the instructor has lost track of which summary belongs to which session. Our task is to recover the original mapping between summaries and sessions, and then build visual and analytical tools that help make sense of the data and enable interaction

Given a collection of jumbled textual summaries stored in an Excel sheet, the goal is to:

- Analyze and preprocess the summaries.
- Reconstruct the mapping between each summary and its original session.
- Visualize overlaps, generate word clouds, and build an application that retrieves top summaries based on keyword input.

Why is This Challenging?

- No labels are available — we don't know which summary belongs to which session.
- Summaries may vary in length, vocabulary, and level of detail.
- Requires careful use of NLP techniques to group and rank summaries.
- The instructor expects a complete data science pipeline, along with a working interactive tool.



SerialNo	Session_Summary
1	we started our lecture with a recap of previous lecture particularly about the difference betw
2	in this session, we explored various feature encoding techniques, essential for converting cate
3	population and sample were further discussed upon. sample is a good and representative par
4	we first looked at all the summaries and observed from the graph that number of people wh
5	midsem metrics for evaluation and also discussion of exercise and feedback on it
6	the lecture explained how we use samples to understand populations. a sample is a small gro
7	in class, we started by talking about how to measure key population parameters like the mea
8	explained logistic regression, which is used for binary classification. we discussed clustering, a
9	we looked at the course summary we initially talked about ways of improving the quality of o
10	the lecture began with a recap of confidence intervals. using an example, we analyzed points
11	problem with heatmap is that they don't capture sufficient information but they play import
12	in today's session, we first analyzed the summaries we write after every class using the pivot
13	first we calculate sample mean,which is considered the approximate of population mean ,the
14	we started off by discussing that with nominal and ordinal data, our observation or data poin
15	in today's lecture, sir discussed steps to solve problem in ds. 1. understanding the problem 2

Cleaning & Preprocessing the Data

To prepare raw summaries for downstream analysis and modeling by standardizing, cleaning, and reducing noise in the text

1. Remove Non-Alphanumeric Characters

- Used re library to eliminate punctuation, numbers, and special characters.
- Preserved only alphabetic characters and whitespace.

2. Tokenization

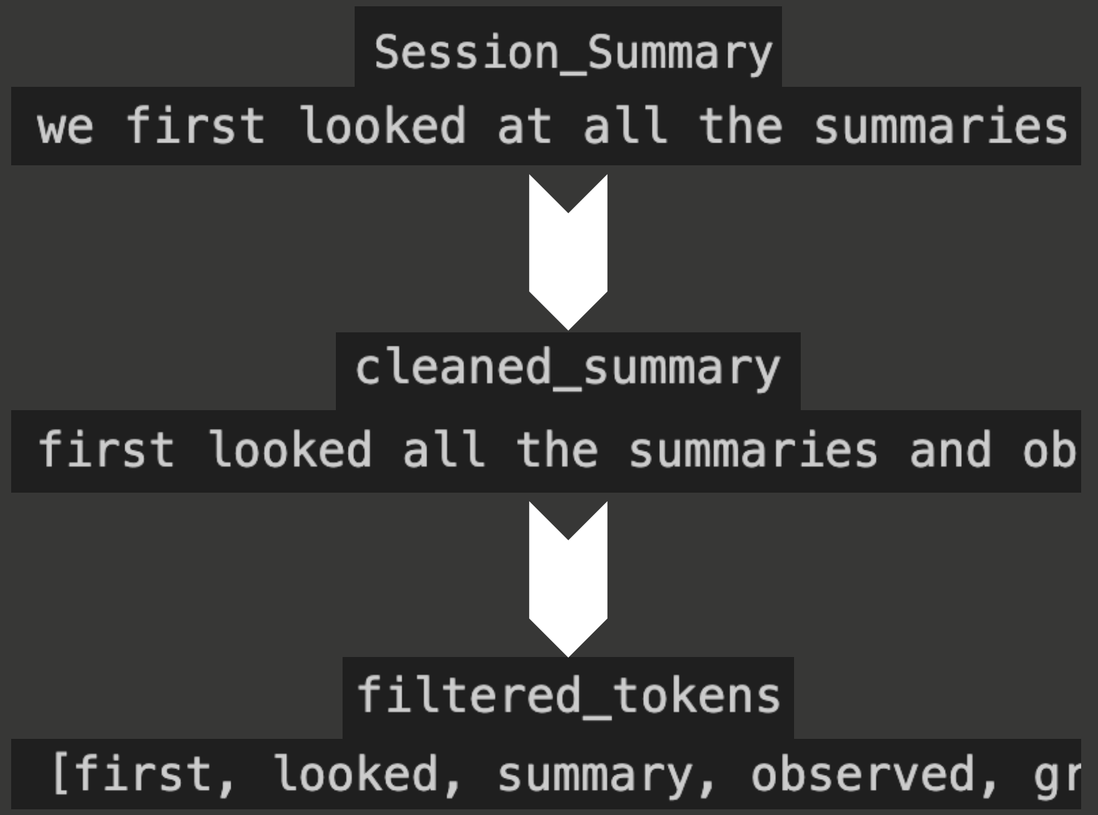
- Split text into individual words (tokens) for analysis.
- Libraries used: nltk.word_tokenize

3. Lemmatization

- Converted each token to its base (dictionary) form.
- More accurate than stemming (e.g., “running” → “run”, “better” → “good”)

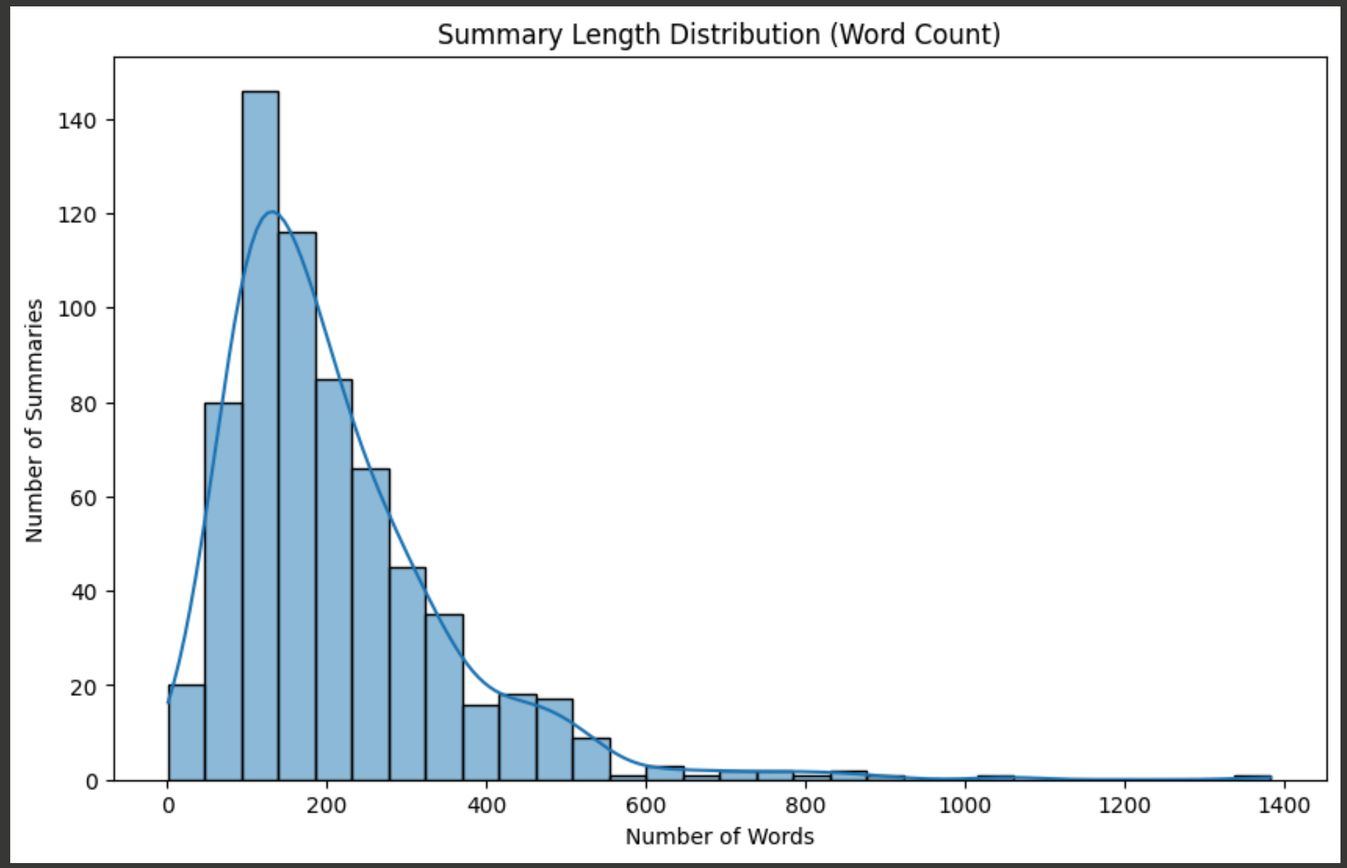
4. Stopword Removal

- Removed common English stopwords like “the”, “is”, “and” to reduce noise



Exploratory Data Analysis

- The plot shows the distribution of summary word counts across the dataset
- Most summaries fall between 100–300 words, peaking near 150 words
- A long tail extends past 600 words, with some summaries exceeding 1000 words
- The distribution is right-skewed, indicating the presence of outlier summaries that are extremely verbose
- High-frequency peak suggests a typical summary length, while the long tail reflects a few detailed or overly lengthy summaries



Top vs. Bottom Summaries

```
Number of unusually short summaries (<10 words): 3

Top 5 Summaries with Most Words:
Summary Index 27: 1384 words
Summary Index 182: 1055 words
Summary Index 529: 894 words
Summary Index 203: 838 words
Summary Index 624: 837 words

Bottom 5 Summaries with Least Words:
Summary Index 20: 1 words
Summary Index 627: 4 words
Summary Index 600: 6 words
Summary Index 24: 12 words
Summary Index 584: 12 words
```

These unusually short summaries could be because of the drive link pasted in the excel data (which act as 1 word) instead of the actual summary

Exploratory Data Analysis

```
Top 10 Frequent Raw Tokens:
[('the', 8524), ('and', 4006), ('data', 2384), ('for', 1403), ('that', 1254), ('which', 1093), ('are', 1031), ('with', 100
```

```
Top 10 Frequent Non-Stopword Tokens:
[('data', 2388), ('model', 1218), ('value', 1047), ('regression', 1004), ('feature', 879), ('sample', 763), ('mean', 669),
```

Top 10 Frequent Raw Tokens

- Includes common stopwords like:
  - 'the' (8524), 'and' (4006), 'for' (1403), 'that' (1254)
- These words occur very frequently but carry little to no contextual meaning, they dominate frequency-based models like Bag of Words or TF-IDF, skewing the representation without contributing to understanding.

Top 10 Frequent Non-Stopword Tokens

- After removing stopwords, more content-rich terms emerge:
  - 'data' (2388), 'model' (1218), 'regression' (1004), 'feature' (879)
- These are relevant domain-specific keywords, reflective of the dataset's focus on: Data analysis, Modeling and Statistical metrics

These tokens will help in Text clustering, Keyword extraction and Content-based recommendation or ranking

- Preprocessing steps like stopword removal significantly improve the interpretability and performance of downstream NLP tasks.
- Shows clear transition from generic language to core domain topics

Exploratory Data Analysis

```
Top word counts for cluster 0:
[262, 219, 151]

Top word counts for cluster 1:
[479, 216, 125]

Top word counts for cluster 2:
[1]

Top word counts for cluster 3:
[138, 128, 104]

Top word counts for cluster 4:
[166, 94, 91]

Top word counts for cluster 5:
[354, 228, 187]
```

```
Top word counts for cluster 6:
[301, 193, 62]

Top word counts for cluster 7:
[154, 140, 116]
...
[350, 127, 79]

Top word counts for cluster 18:
[505, 362, 335]
```

Our First Approach

Defining Our Own Corpus

- We built a custom vocabulary from all the unique words present across session summaries
- This forms the foundation for our Bag of Words (BoW) representation.

Embedding Technique: Bag of Words

- Each summary is converted into a fixed-length vector based on word frequencies.
- No semantic understanding — purely based on word occurrence.
- Ignores context, word order, and meaning

From Words → Sentences → Paragraphs

- We aggregate all word vectors in a summary to form one document vector.
- Final representation is a high-dimensional sparse vector.

Distance-Based Objective Function

- We use Euclidean distance between vectors to measure similarity or closeness.
- Closely located vectors → similar sessions.

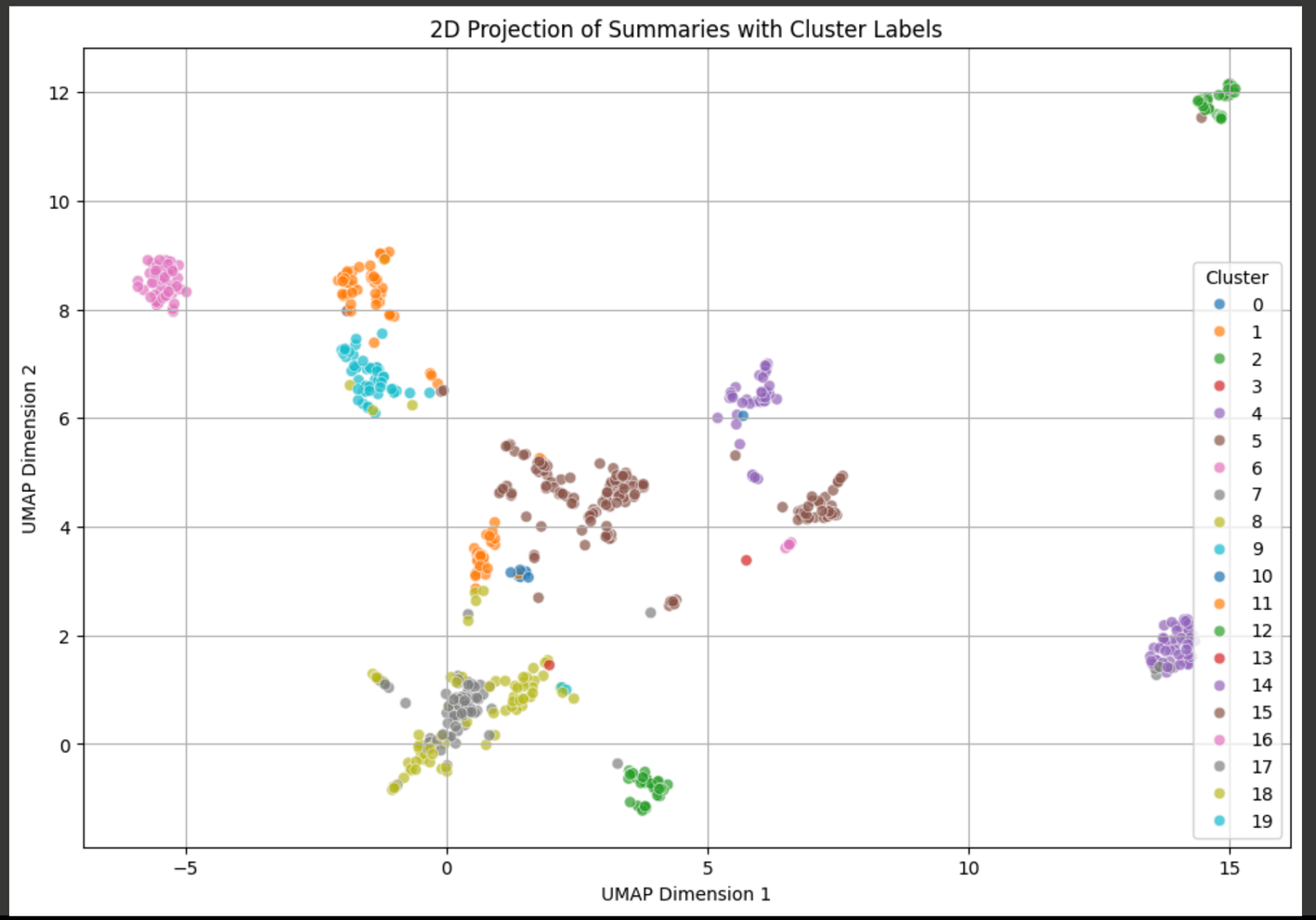
Implementing Model

Clustering with UMAP Projection

- We applied UMAP (Uniform Manifold Approximation and Projection) to reduce high-dimensional vector representations into 2D for visualization.
- This dimensionality reduction helps in visualizing the separation and structure of clusters.

The plot shows distinct clusters, each color representing a different group of summaries. Clear separation suggests that our textual representations (even with simple embeddings) capture some inherent structure in the data. Some clusters are very compact, indicating strong similarity within that group. A few inter-cluster overlaps suggest possible semantic similarity or limitations in embedding/model

This visualization validates our clustering pipeline. Shows that unsupervised learning methods like clustering can work effectively, even without labels, when combined with good preprocessing and embedding strategies.Acts as an intermediate evaluation before moving to more complex or semantic embedding models.



Analysing the Results

- The Silhouette Score is 0.0442, which is very low, indicating that the clusters formed are not well separated and there's significant overlap in the feature space.
- This suggests that the current embeddings (likely based on a bag-of-words or similar non-contextual method) are not capturing semantic meaning effectively.
- The table on the right shows summaries assigned to clusters, but with similar tokens appearing across clusters (e.g., “lecture”, “sample”, “discussion”), reinforcing that the model struggles to distinguish semantically different summaries.
- In short: the clusters lack coherence, which points to a need for more semantically rich representations (like contextual embeddings).

Silhouette Score: 0.0442

Here are the results of cluster sumamries

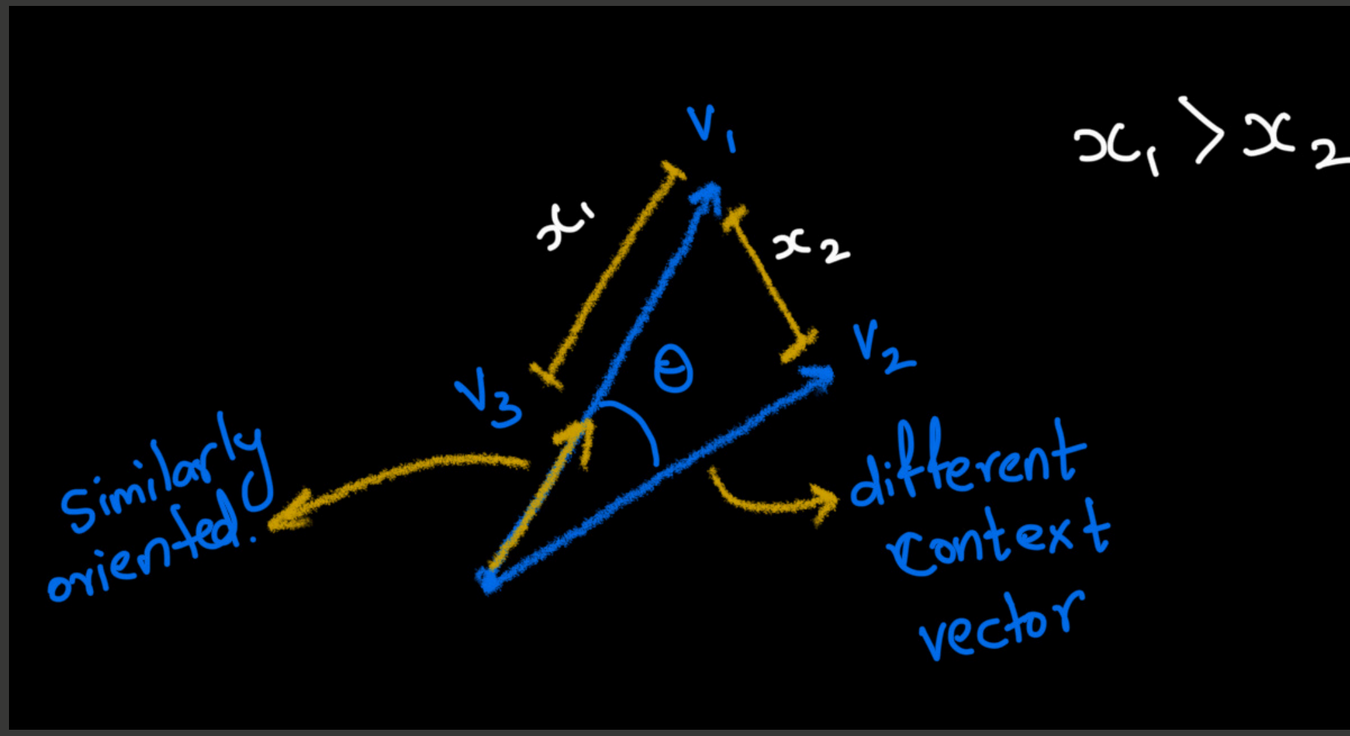
	filtered_tokens	cluster
0	[started, lecture, recap, previous, lecture, p...	6
1	[session, explored, various, feature, encoding...	8
2	[population, sample, discussed, upon, sample, ...	3
3	[first, looked, summary, observed, graph, numb...	6
4	[midsem, metric, evaluation, also, discussion,...	12
..	...	...
663	[lecture, learnt, population, sample, majorly,...	3
664	[today, continued, discussion, statistical, si...	11
665	[beginning, sir, explained, way, determine, nu...	16
666	[studied, crispdm, cross, industry, standard, ...	10
667	[today, class, interpreted, pvalues, confidenc...	17

Why will it not work?

- The clustering metric used here is Euclidean distance, which is sensitive to the magnitude (length) of the vectors rather than just their orientation (meaning)

In the diagram:

- $x_1 > x_2$ means the vector on the left is longer than the one on the right.
- Even though v_1 and v_2 point in different directions (i.e., they have different contexts), their Euclidean distance is smaller than that between v_1 and v_3, just because x_1 is longer.
- This is a problem: two vectors with completely different meanings but longer lengths might get grouped together more than two similarly-oriented vectors of smaller magnitude.
- In essence, Euclidean distance penalizes short vectors even if they represent similar content—leading to semantically incorrect clusters.



- ✅ A better alternative: Use cosine similarity, which considers orientation rather than length, and is much better suited for comparing embeddings that represent meaning.

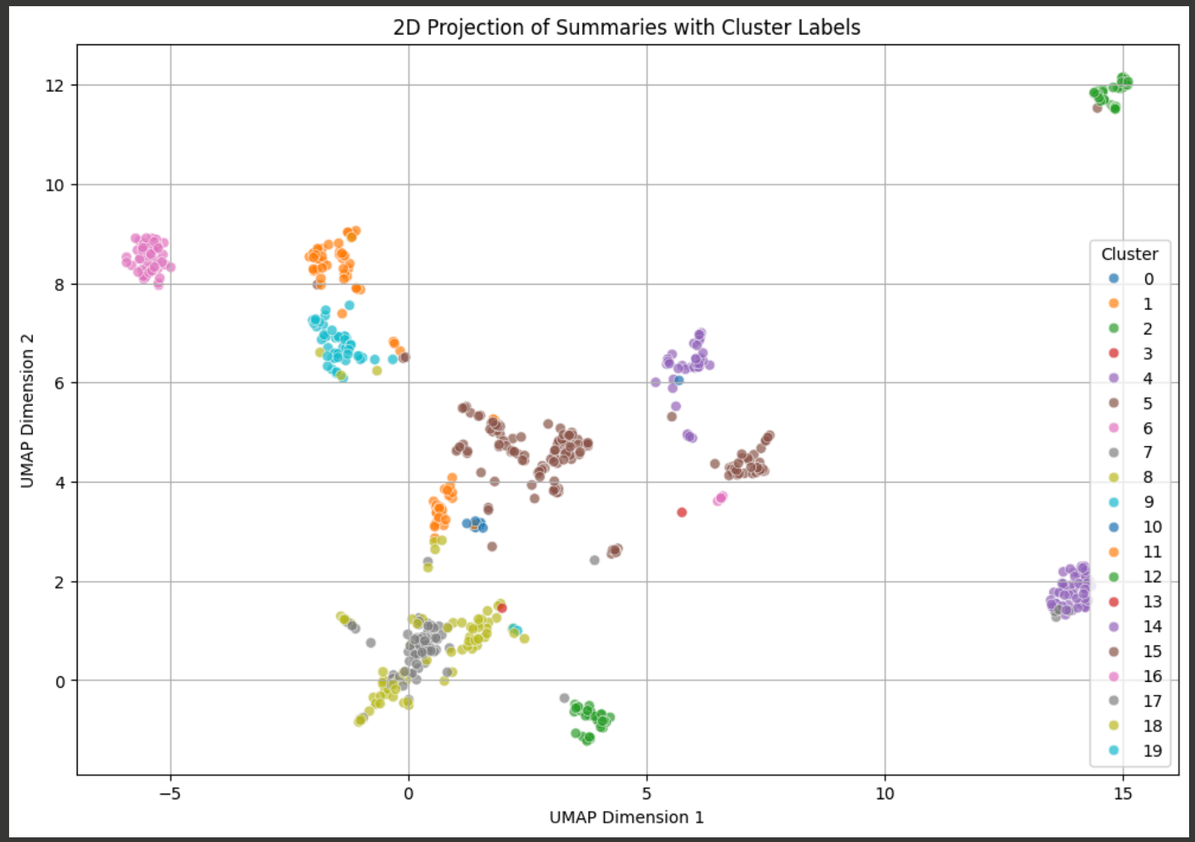
Tweaking the Model

Using Cosine Similarity instead of Euclidean Distance: `Silhouette Score: 0.0609`

- After switching to Cosine Similarity, we observe:
  - Clearer boundaries between some clusters (visual improvement in UMAP).
  - But still a low Silhouette Score: 0.0609

→ This implies poor separation between clusters overall.

- Why is this happening?
  - Possible reasons:
    - The summaries are too short or noisy, limiting meaningful clustering.
    - UMAP + cosine might still not capture topic continuity well.
    - Clustering on sentence embeddings might need better preprocessing or dimensional tuning.



Why Semantic meaning should be considered?

Clustering based on raw word occurrences misses the actual meaning of the content. It simply groups texts that have overlapping vocabulary — not those that are talking about the same idea.

Semantic meaning is not captured in the current approach. As a result, clustering is usually based on similar types of word occurrence.

For example:
If one person writes the same word many times in one session, and another person uses those same words repeatedly in a different session — they will likely be categorized into the same cluster, even though they're talking about different things.
Also, if two people describe the same concept using different sets of words in the same session, they may be assigned to different clusters.

This leads to low precision in clustering.
→ That's why we need to move towards semantic embeddings, which consider meaning, not just word frequency.

Moving towards Word2Vec

Word2Vec for capturing the semantic meaning of text. Unlike traditional models that rely purely on word frequency or surface-level similarity, Word2Vec generates dense vector representations that reflect contextual relationships between words. This allows us to embed entire sentences based on the meanings of their words, enabling better grouping of similar content even when different vocabulary is used. By averaging word vectors, we obtain a robust sentence-level embedding that significantly improves clustering precision by focusing on meaning rather than mere word overlap.

Word2Vec is implemented using the Gensim library to generate semantic embeddings for text data. It begins by converting tokenized sentences into a list format and then trains a Word2Vec model on them. A custom function sentence_to_vector is defined to convert each sentence into a vector by averaging the vectors of the individual words present in the model's vocabulary. If no known words are found, it returns a zero vector. Finally, the embeddings are applied to each row in the DataFrame using a lambda function, storing the results in a new column word2vec_embedding.

Embedding with semantic meaning being captured.(Word2vec)

```
[112] sentences = df['filtered_tokens'].tolist()  # Convert tokenized sentences into list format
      w2v_model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

      def sentence_to_vector(tokens, model, vector_size):
          vectors = [model.wv[word] for word in tokens if word in model.wv]
          if len(vectors) == 0:
              return np.zeros(vector_size)
          return np.mean(vectors, axis=0)

      df['word2vec_embedding'] = df['filtered_tokens'].apply(lambda tokens: sentence_to_vector(tokens, w2v_model, 100))
```

Moving towards Word2Vec

- Sentence embeddings created using the Word2Vec model.
- Converts text into dense 100-dimensional vectors.
- Captures semantic meaning while reducing dimensionality.
- Enables efficient categorization and downstream analysis.
- Displayed output shows vector for one sentence post-embedding

```
df['word2vec_embedding'][0]
```

```
array([-0.28186896,  0.30858818,  0.15069692,  0.011706  ,  0.22381249,
       -0.5119489 ,  0.19180134,  0.992887  , -0.52338403, -0.26764652,
       -0.19004942, -0.62850136, -0.02465848,  0.1202359 ,  0.30252993,
       -0.25060356,  0.13352409, -0.3175496 ,  0.0297188 , -0.57478076,
        0.49670848,  0.31892055,  0.16547517, -0.33597597, -0.13601463,
       -0.03241756, -0.31000954, -0.41030034, -0.25861832, -0.02409971,
        0.443491  , -0.09199211,  0.22437775, -0.25124106, -0.15432476,
        0.56490195,  0.16470434, -0.12530173, -0.2666285 , -0.6574196 ,
        0.13630295, -0.46547672, -0.37360957,  0.08245689,  0.22470146,
       -0.13049458, -0.3905026 , -0.0615961 ,  0.16592063,  0.35083166,
        0.08823225, -0.41117933, -0.1315094 ,  0.04913234, -0.12757103,
        0.4052682 ,  0.25057307, -0.06000089, -0.2580284 ,  0.2738212 ,
       -0.07349873,  0.05022931,  0.05330835, -0.25477308, -0.5521576 ,
        0.19016917,  0.07869132,  0.27264732, -0.5454135 ,  0.5149326 ,
       -0.3658546 ,  0.0710806 ,  0.39461833, -0.17952567,  0.2894197 ,
        0.09786405,  0.03819945, -0.00553137, -0.3883687 ,  0.24494451,
       -0.1802477 ,  0.01504766, -0.38748142,  0.59291685, -0.10700724,
        0.12253742, -0.01420076,  0.47962162,  0.6389808 ,  0.1134556 ,
        0.55148196,  0.20043023,  0.08871664,  0.26942146,  0.64366704,
        0.34603617,  0.06106414, -0.3704791 ,  0.25676847, -0.08656558],
      dtype=float32)
```

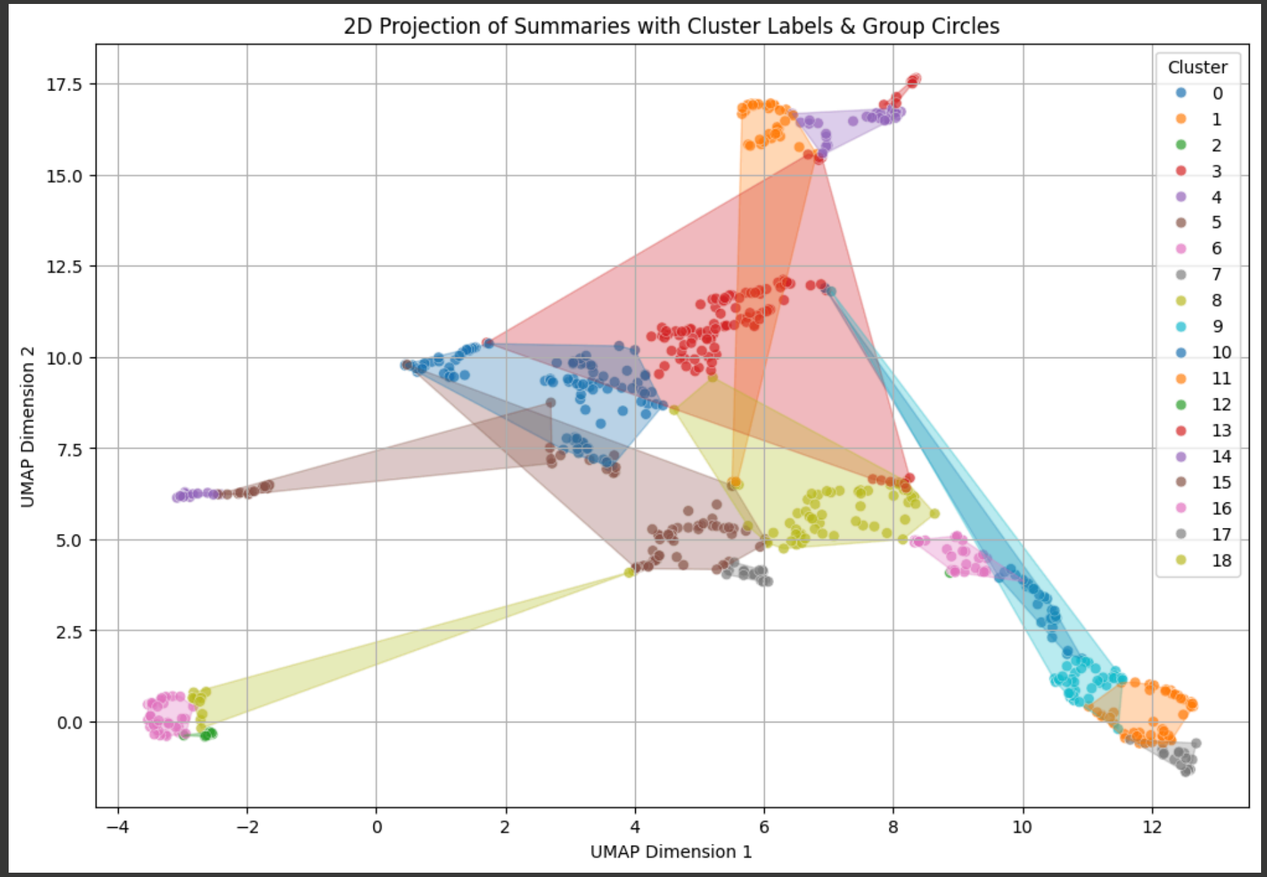
Analysing the Results

```
Silhouette Score: 0.2691
```

```
sil_score = silhouette_score(X, df['cluster_wv'])
print(f"Silhouette Score: {sil_score:.4f}")
print(df[['filtered_tokens', 'cluster_wv']])
```

```
Silhouette Score: 0.2691
```

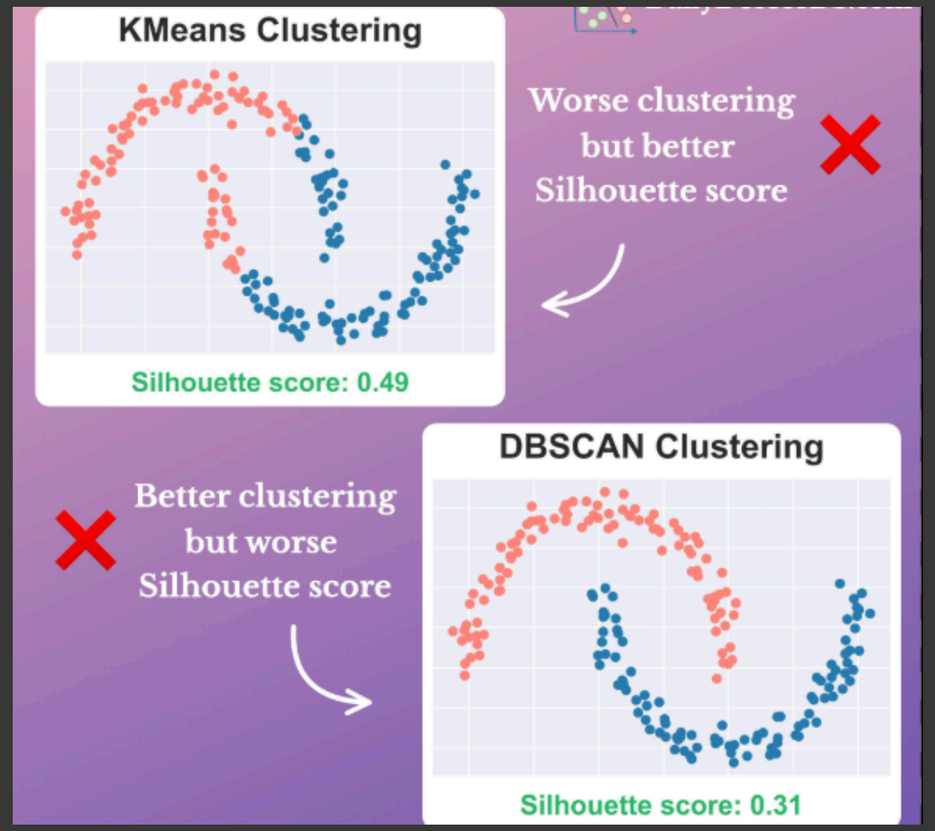
- UMAP projection shows better grouping compared to earlier models.
- Improved Silhouette score = 0.2691 → Indicates some cluster structure, but still weak.
- Certain clusters show clear separation; others have significant overlap.
- Outliers are present, impacting overall cluster integrity.
- Classification basis may not be entirely reliable due to semantic ambiguity



Factors which Silhouette Score doesnt take into account

Limitations of Silhouette Score

- Silhouette Score measures cluster cohesion and separation — not semantic correctness.
- A high score doesn't guarantee meaningful classification.
- Example: If clustering is based on vocabulary overlap (e.g., student summaries),
- similar word usage may inflate the score — even if interpretations differ.
- Possible misclassification: grouping students with similar vocabulary,
- rather than clustering by actual thematic content or intent.
- Hence, evaluation must consider both statistical scores and contextual relevance.



How TF-IDF is better embedding?

- TF-IDF (Term Frequency-Inverse Document Frequency) adjusts word importance based on both intra-document repetition and inter-document relevance.
- Words that appear too frequently within a single summary are down-weighted, preventing redundant or filler terms from dominating the embedding vector.
- This is particularly useful in student-generated summaries: a student with low retention might repeat basic words to artificially extend the summary. TF-IDF corrects this by reducing the weight of those frequently repeated terms.
- Meanwhile, terms that are shared across multiple summaries (and used meaningfully) are up-weighted — capturing cross-summary semantic relevance.
- Unlike simple frequency-based embeddings, TF-IDF better captures contextual depth and helps distinguish between summaries that are genuinely comprehensive versus those that are repetitive.
- As a result, vector representations become more aligned with actual content quality rather than superficial length or word overlap — which significantly improves clustering accuracy.

Why a single featurization will fail?

- Early models like Bag of Words or basic TF-IDF rely only on word frequency and ignore semantic context.

→ These models group summaries based on surface-level word overlap, not meaning.

- As we've seen, incorporating semantic awareness drastically improves clustering quality.

→ Simply defining a custom corpus and applying frequency-based techniques improves scores, but still misses context.

- Even TF-IDF alone may fail in scenarios where:
  - Different words convey the same idea (e.g., "jobless" vs. "unemployed")
  - Students use varied vocabulary to describe the same session

- Therefore, relying on a single featurization method — whether frequency-based or semantic — is insufficient.

- What we need is a combination: TF-IDF to emphasize meaningful words + semantic embeddings (like Word2Vec) to capture contextual meaning

Incorporating Word2Vec with TF-IDF

Incorporating Word2Vec with TF-IDF

- TF-IDF alone helps by reducing the weight of overused or filler words in summaries — correcting for superficial length or repetition.
- However, it still does not capture the true semantic context or intent behind the words

Why This Matters in Our Case:

- A student with lower retention might use the same word repeatedly to artificially extend the summary.

→ This repetition skews the vector representation, making it appear different from summaries that convey real understanding.

- A sincere student, focusing on the lecture's flow, may write a well-structured and context-rich summary with varied vocabulary.

→ That summary might look very different vectorially, even though it's more accurate

The Solution: Combine TF-IDF with Word2Vec

- Word2Vec captures the semantic meaning of words and their relationships.
- TF-IDF ensures that repetitive or generic words don't dominate the embedding.
- By weighting Word2Vec embeddings with TF-IDF scores, we:
  - Preserve semantic context
  - De-emphasize noisy, overused tokens
  - Generate fairer, more meaningful sentence-level representations

→ This combination results in more reliable clustering and analysis of summaries across different writing styles.

Creating Hybrid Models

Why Hybrid TF-IDF × Word2Vec Works Better

- TF-IDF gives us the importance of words relative to all summaries
- Word2Vec captures the semantic relationship between words
- Combining both gives us an embedding that is:
  - Context-aware
  - Importance-weighted
  - Less biased by writing style or filler content

Key Steps in Hybrid Embedding:

1. Preprocess summaries → Filtered tokens
2. Compute TF-IDF vector for each summary
3. For each word:
   - Check if it exists in both TF-IDF and Word2Vec vocab
   - Multiply Word2Vec vector by its TF-IDF score
4. Normalize and average to get final embedding vector

What's the Impact?

- Reduces influence of filler or repeated words
- Gives fairer representation of summary's actual content
- Greatly improves clustering and classification outcomes

Cosine tf-idf

```python
# Prepare TF-IDF
df['text_for_tfidf'] = df['filtered_tokens'].apply(' '.join)
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df['text_for_tfidf'])

# Create vocabulary mapping
index_to_word = {idx: word for word, idx in tfidf_vectorizer.vocabulary_.items()}

def hybrid_embedding(tokens, model, tfidf_row, vector_size=100):
    vectors = []
    weights = []

    # Get TF-IDF scores for document words
    for idx, score in zip(tfidf_row.indices, tfidf_row.data):
        word = index_to_word[idx]
        if word in tokens and word in model.wv:
            vectors.append(model.wv[word])
            weights.append(score)

    if not vectors:
        return np.zeros(vector_size)

    # Normalize weights and compute weighted average
    weights = np.array(weights) / sum(weights)
    return np.dot(weights.T, vectors)

# Apply to create hybrid embeddings
df['hybrid_embedding'] = [
    hybrid_embedding(row['filtered_tokens'], w2v_model, tfidf_matrix[i], 100)
    for i, row in df.iterrows()
]
```

Why Deep Learning wont be effective here?

Heavy for the Task

- Requires massive compute & time to train
- Overkill for small or noisy datasets like student summaries

Not Focused on What We Want

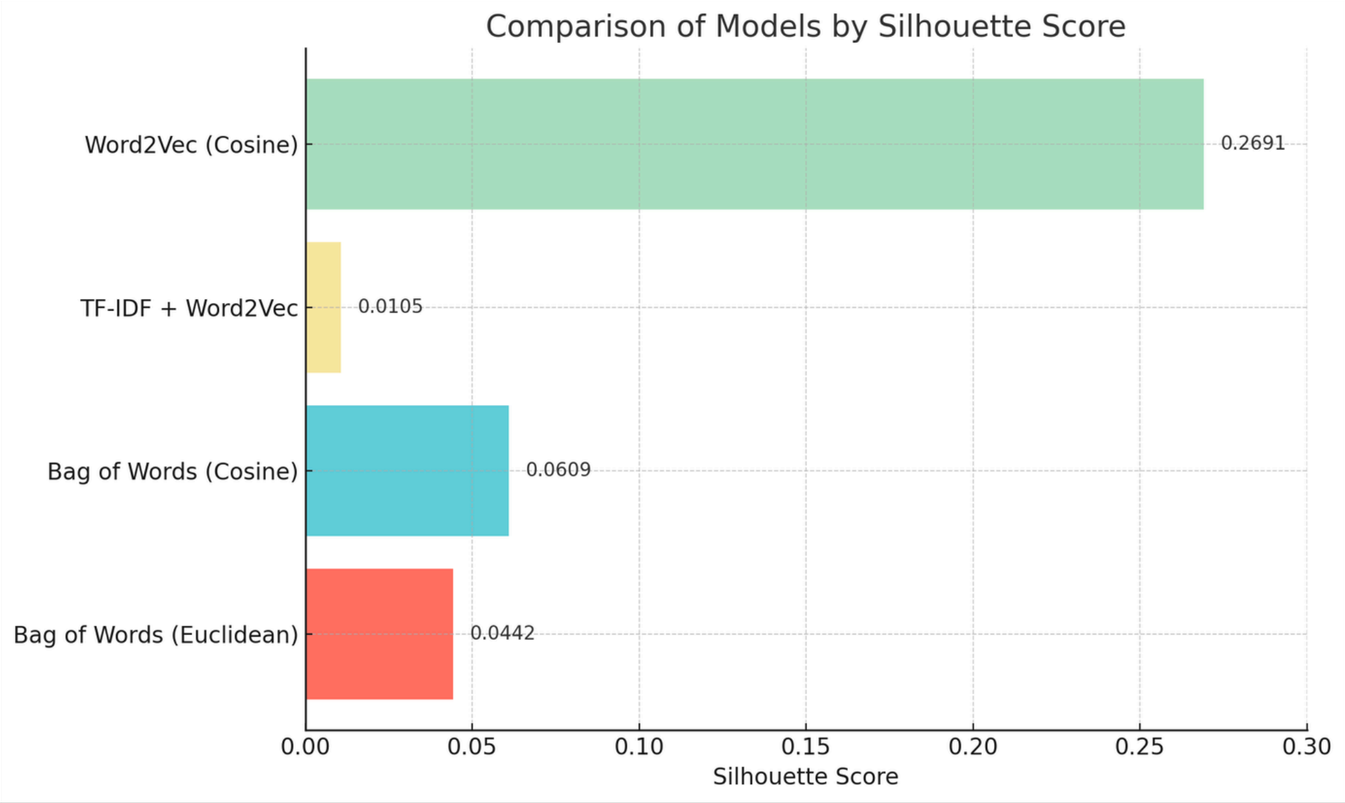
- Models like BERT focus heavily on grammar and syntax
- May cluster summaries based on writing style instead of content
- Risks favoring well-written or “fancy” English over actual context

Overfitting Risk

- Small variations in vocabulary can confuse the model
- Tends to memorize rather than generalize meaning in small datasets

✅ Instead, lightweight hybrid methods give more control and interpretability

Comparing Various Models



While silhouette scores offer a quantitative metric for clustering evaluation, in our context they fail to reflect true semantic grouping.
For example:

- A high score might come from clustering based on writing style or vocabulary, rather than conceptual understanding.
- TF-IDF + Word2Vec, despite being semantically aligned, shows a low silhouette score due to vocabulary overlap or abstract structure.

Silhouette score is not a sufficient standalone metric for this task. We supplemented it with manual validation using actual student summaries to verify semantic clustering

Documenting our Summaries for Validation

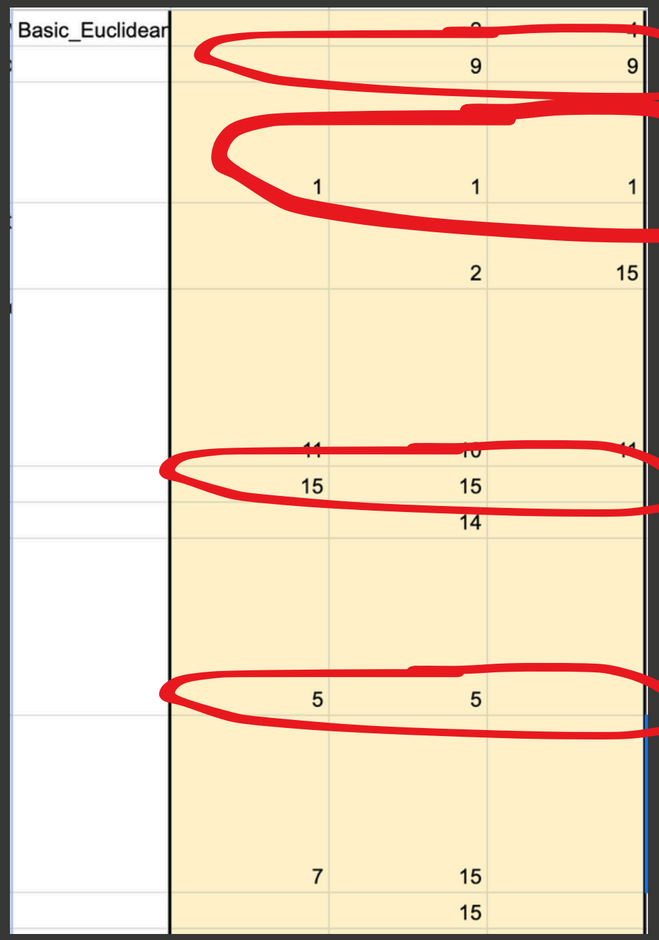
These are the summaries of 3 students that we got and we arranged them based on the dates and then ran different models on them

Since we cannot classify on the basis of Silhouette score we need some metric to judge our model, hence we are using our summaries, **if summaries written on same date are in same cluster we would add score to the model**

Wed 15 Jan		Learned about four levels of measurement in statistics: Nominal, Ordin	Class 3 summary: First of all we talked about how to fit a curv
Fri 22/01		We did linear regression on a dataset in Excel. Calculated beta0 and be	we first compared about the estimators being the statistics c
Fri 24/01	Thr best way to make use of a single sample (with Multiple Linear Regression. Anova. Statistical equivalence of multiple aberage F-static, MSR over MSE.	Continued statistical concepts from earlier. Explored data analysis term	We continued in this lecture defining the various result tools.
Wed 29/01		Recap on confidence interval. Outlier from lower probability region is i	In this class first we did the recap for the p-value and the stat
Fri 31/01	Train vs Test (unseen) data split for any ML model. Loss of degree of freedom in R-squared. When on Linear regression need not output a line as the rel sklearn or SciKitLearn has many builtin functions i The notion of better models comes from comes fr Different tests for close to normal distribution. Sk	We recapped in the starting about the closed form solution in multiple	Class 8, Explained how multicollinearity can be a problem an
Fri 5/2/25	Linear regression is expressed as a sum of linearly	We saw there are two methods of improving quality of results: improv	
Wed 7/2		We again started our discussion of clustering and logistic regression. Lo	
Tue 12/2	KNNs and varying behaviour with changing feature Introduction TNR, TPR (aka Specficity, sensitivity), K-Means algorithm and hierarchial clusteting. Exp KNNs and varying behaviour with changing featur Introduction TNR, TPR (aka Specficity, sensitivity), K-Means algorithm and hierarchial clusteting. Exp	We revised score, precision, recall, confusion matrix, accuracy and also	
Fri 14/2/25	How to deal with bad data? CrISPDM asa a cyclical A few observations were made about the features In a pollutants dataset, three types of missing valu DBScan is a clustering method is similar to Kmean HW: why is median preffered over mean when ide Caution! Not every EDA methods needs yo be app	We started session with discussion of exploring data basically EDA (not	
Wed 19/2/25		In this class, much was not done. We moved further in explorator	

Results

Model Self-defined corpus with euclidean objective function in clustering.



They all are classified into same cluster which is a good sign

Using the condition explained in the previous slide we have done the mapping of the three students summaries here

Results

Doing it for all the models

Basic cosine			
		13	14
		0	0
	1	1	1
		9	6
	9	6	9
	17	17	
		18	
	2	18	
	3	17	
		6	

Word2Vec_eucli			
		1	13
		3	0
	16	16	16
		15	9
	13	9	1
	16	16	
		4	
	5	18	
	9	16	
		9	

Word tovec cosi			
		19	14
		5	5
	5	5	14
		3	6
	6	9	3
	6	11	
		16	
	0	13	
	10	6	
		6	

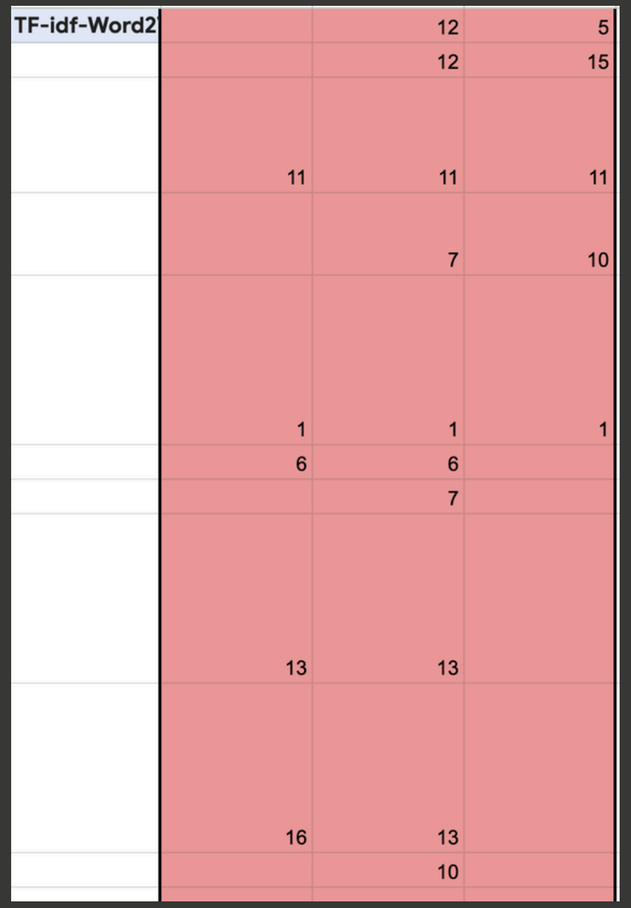
TF-idf-Word2			
		12	5
		12	15
	11	11	11
		7	10
	1	1	1
	6	6	
		7	
	13	13	
	16	13	
		10	

Results

Why manual validation was essential?

- Automated metrics like Silhouette Score gave misleading results due to the subjective nature of our summaries.
- Manual inspection revealed clusters in TF-IDF + Word2Vec were semantically consistent.
- Students with similar understanding levels grouped together more clearly in the hybrid model

Even without accuracy metrics, qualitative review clearly highlighted the superiority of our hybrid model.
This method aligns better with how real understanding manifests in language

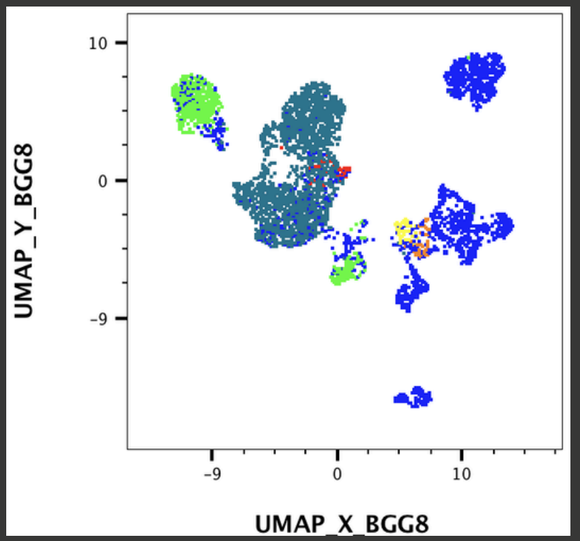


TF-idf-Word2			
		12	5
		12	15
	11	11	11
		7	10
	1	1	1
	6	6	
		7	
	13	13	
	16	13	
		10	

Understanding UMap for Visualization

UMAP (Uniform Manifold Approximation and Projection) is a powerful non-linear dimensionality reduction technique often used for visualizing high-dimensional data in 2D or 3D

- **n_neighbors**: Defines the number of nearest neighbors UMAP considers for each point. Controls the balance between local and global structure in the data. Smaller values → focus on local structure (fine clusters). Larger values → preserve more global relationships.
- **min_dist**: Minimum distance between points in the low-dimensional space. Controls how tightly points are packed. Lower values → tighter clusters, potentially more separation. Higher values → more spread-out clusters, sacrificing compactness.
- **Method of Projection:** UMAP uses:
  - Graph Construction in high dimensions via fuzzy simplicial sets.
  - Optimization in low dimensions by minimizing the cross-entropy between the fuzzy set representations in high and low dimensions



UMAP builds a weighted k-nearest neighbor graph (fuzzy simplicial complex) in the original space, then tries to preserve this structure in lower dimensions by optimizing:

$$\text{Loss} = \sum_{i \neq j} \left[P_{ij} \log\left(\frac{P_{ij}}{Q_{ij}}\right) + (1 - P_{ij}) \log\left(\frac{1 - P_{ij}}{1 - Q_{ij}}\right) \right]$$

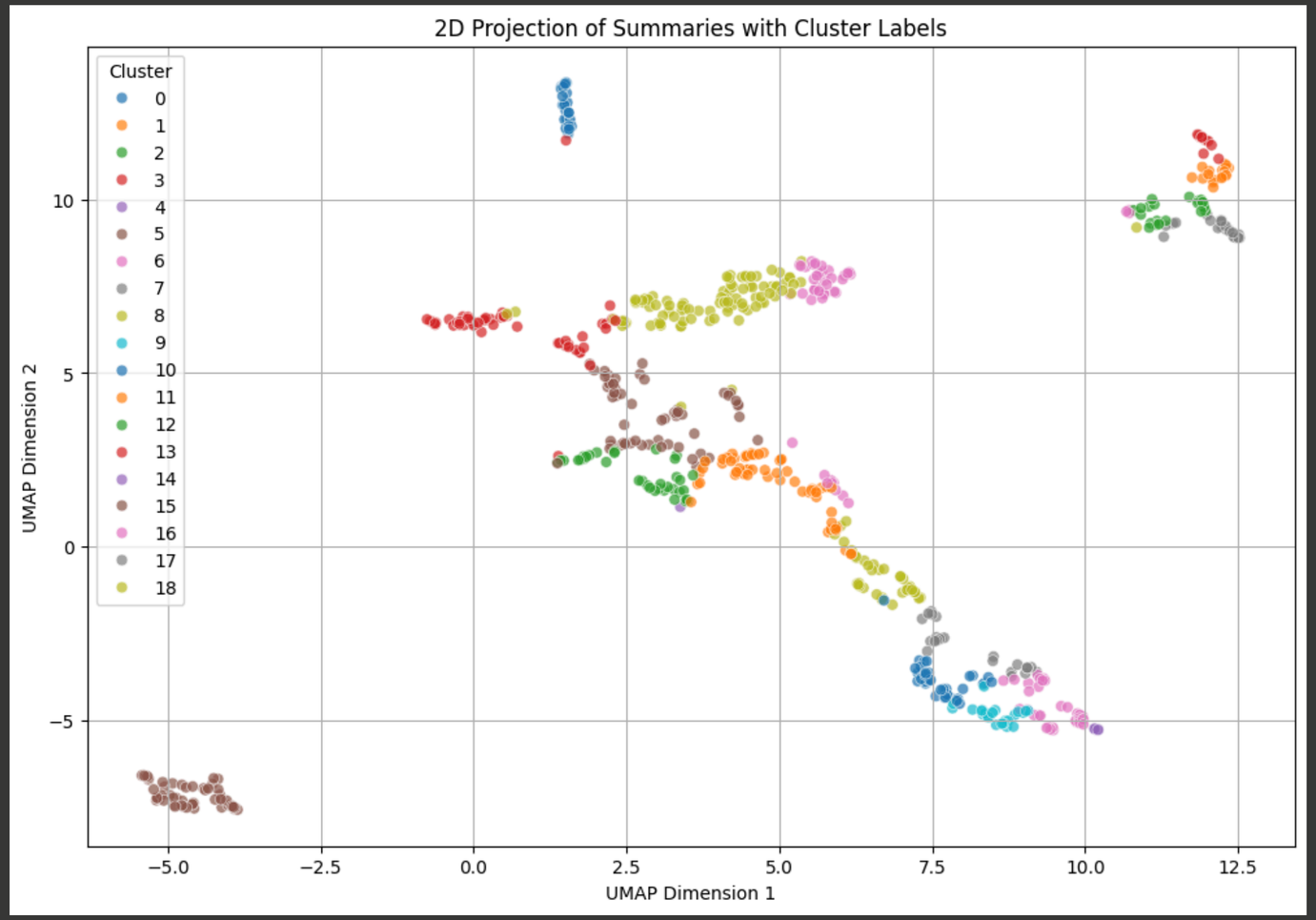
Where:

- Pij is the probability of similarity in high-dimensional space.
- Qij is the probability in low-dimensional space.

This is a form of cross-entropy loss between fuzzy sets

Summaries Visualization

This UMAP projection gives a clear, low-dimensional snapshot of how our generated summaries are distributed and clustered. Each color represents a different cluster label assigned during the unsupervised learning phase

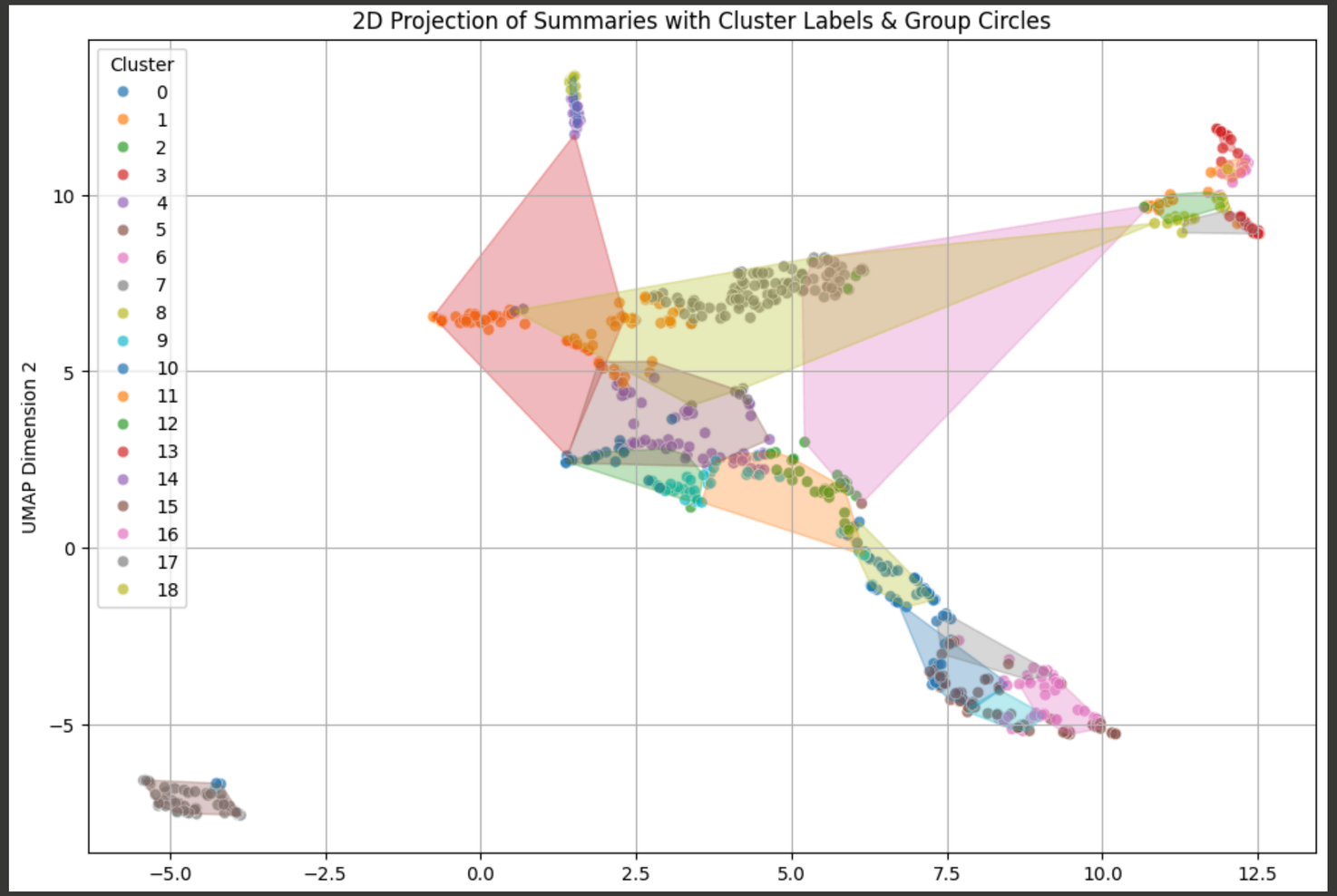


Summaries Visualization

Key Observations:

- Well-separated Clusters: Most clusters are spatially distinct, indicating that the summaries within each group share strong semantic similarities. This supports that our feature representation captures meaningful linguistic patterns.
- Minimal Overlap: The minimal intersection between clusters suggests that the data points are well-differentiated, which is promising for downstream applications like classification or recommendation.
- Smooth Gradient-like Flow: There's a visible flow in some regions, especially where the clusters gently transition. This hints at possible topic drift or thematic progression in the corpus
- Some clusters such as 1, 2, and 6 lie within the same region, hinting that while they were separated during clustering, they might share overarching topical similarities.
- Others, like clusters 0 or 16, are completely isolated—these could correspond to highly specialized or outlier summaries.
- The grouping of clusters provides a richer semantic map and may aid in downstream manual annotation or taxonomy formation.

When validating clusters qualitatively (like with summary reading), this structure helps prioritize which groups to treat together and which demand individual attention. It adds an interpretability layer to the unsupervised pipeline



Why a clear separation is not possible?

While our clustering and visualization models aim to objectively separate summaries, the nature of human comprehension introduces subjectivity

Some students paraphrase in their own words and natural tone, while others mimic the speaker's phrasing. This causes similar content to appear linguistically distant

Students not paying full attention tend to summarize broadly—often including generic remarks that apply to multiple sessions. Conversely, highly focused students may capture niche points, leading to summaries that look unique—even if the underlying session content overlaps.

This inconsistency introduces noise in embeddings, making clusters less clean or intuitive. It highlights the limitation of relying solely on text representation without human-in-the-loop understanding.

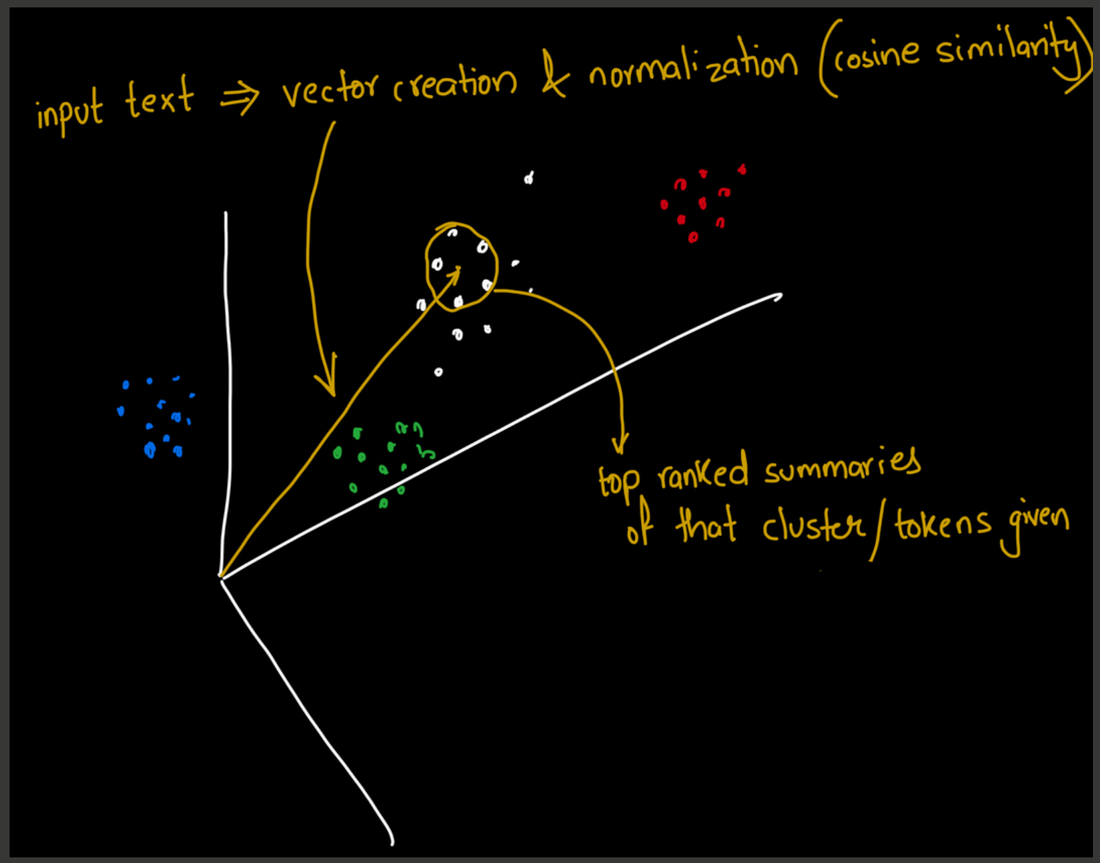
Clustering isn't failing—it's revealing how diverse student perceptions can be. This reinforces the need for manual validation when interpreting clusters.

How will the summaries be ranked?

We convert all summaries into vectors using NLP techniques.

- These vectors are grouped into clusters based on similarity (topics, tone, content).
- Each cluster has a center point - called the cluster centroid.
- Summaries closest to that centroid (based on cosine similarity) are ranked higher

The more “on-topic” and representative your summary is for that cluster, the higher it will be ranked



Ranking of Summaries for each session

```
keywords_list=['pca']
```

```
🔍 Most relevant session cluster for keywords ['pca']: Cluster 3
```

	Session_Summary	similarity_to_keyword
615	focused on principal component analysis (pca) ...	0.990864
574	the session highlighted the use of principal c...	0.990700
99	today covered principal component analysis (pc...	0.989237

```
keywords_list=['midsem','summary','today','class']
```

```
🔍 Most relevant session cluster for keywords ['midsem', 'summary', 'today', 'class']: Cluster 3
```

	Session_Summary	similarity_to_keyword
627	paper discussion of midsem	0.999206
408	we discussed mid sem exam and assignment 3 and...	0.997502
76	the session focused on essential data analysis...	0.997486

These examples illustrate how we can retrieve and rank the most relevant summaries from a session using keyword-based clustering.

- In the first case, the keyword ['pca'] returns summaries that heavily discuss Principal Component Analysis — all with very high similarity scores (~0.99), meaning the summaries are tightly aligned with the search intent.
- The second example uses broader keywords like ['midsem', 'summary', 'today', 'class'], and the model still identifies the most relevant summaries from the session. Despite the generic nature of the keywords, the clustering captures contextual relevance really well.

✅ Takeaway: This approach is flexible — it works for both niche technical topics and general queries — and ranks results based on their actual content alignment rather than just keyword matchin

Word Cloud Algorithm

A word cloud is a visual representation of the most frequent words in a body of text. It highlights words that appear more often by increasing their size or boldness in the visualization

Word clouds help identify dominant themes or keywords from large textual data by using word frequency or weighted scores (like TF-IDF) to determine prominence

To assign weights to words, we often use the Term Frequency-Inverse Document Frequency (TF-IDF):

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Where:

- $\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in } d}$
- $\text{IDF}(t, D) = \log\left(\frac{N}{1+\text{df}(t)}\right)$

N = total number of documents

$\text{df}(t)$ = number of documents containing the term t

Words with higher TF-IDF scores appear larger in the word cloud — helping us visually grasp what terms were most distinctive or important

WordCloud of all Summaries

The word cloud shows “class” and “model” as the most prominent terms — and that’s intuitive:
“Class” appears frequently because:

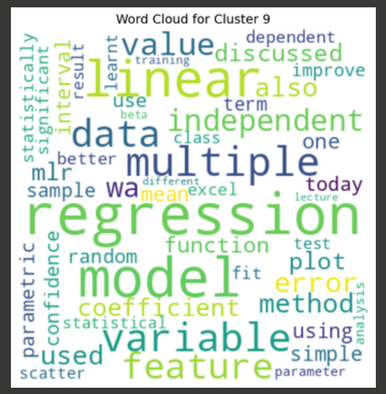
- It refers to both the lecture sessions (e.g., “in today’s class...”) and classification tasks in ML (e.g., “positive class”, “binary class”).
- It's a naturally recurring word when students summarize discussions

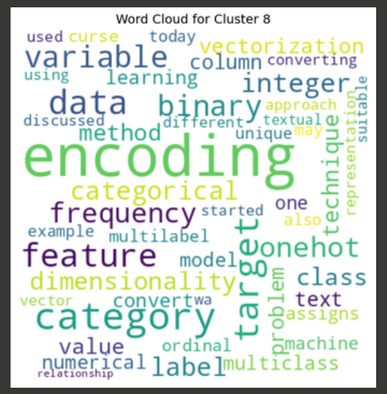
“Model” dominates because:

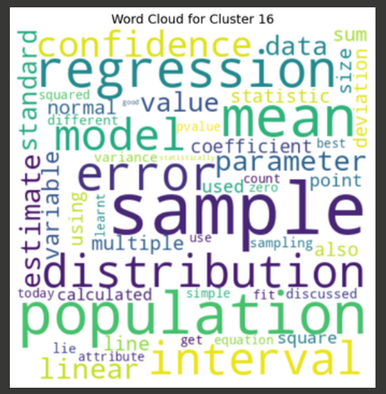
- Machine learning sessions often revolve around different models — logistic regression, decision trees, SVM, etc.
- Most sessions involve training, testing, or evaluating a model — making it an anchor word in technical summaries

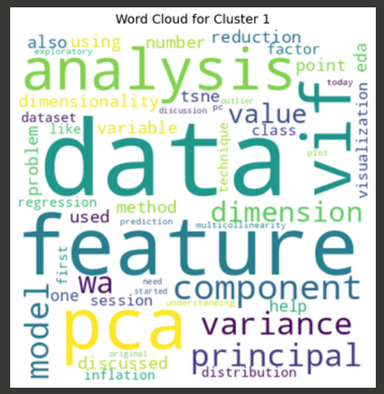
So, their frequency isn’t just noise — it reflects the backbone of what was taught and discussed across sessions.

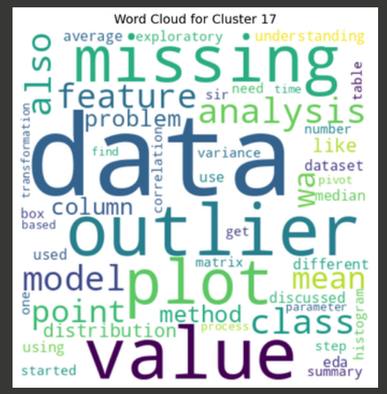
Each Session Word Cloud

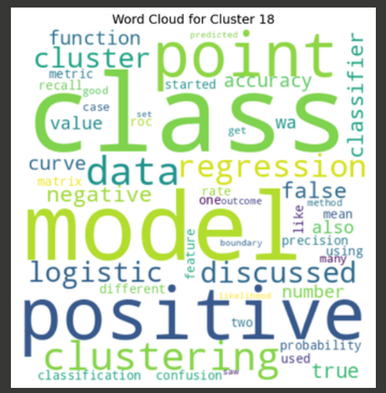


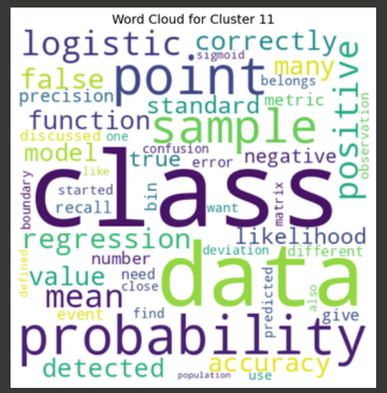


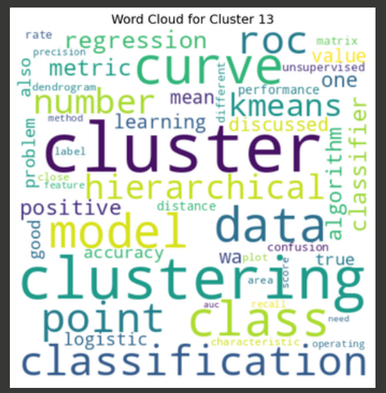




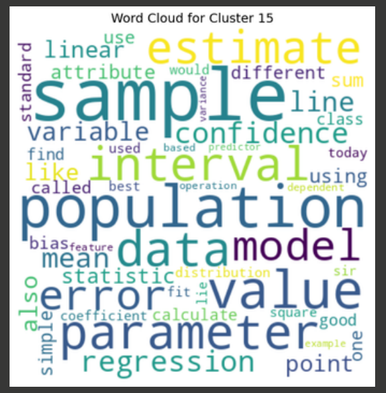


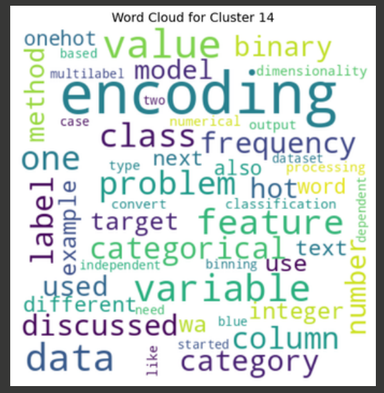


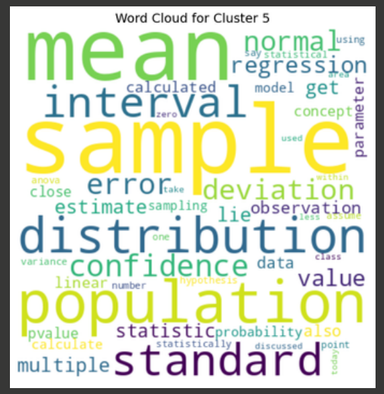


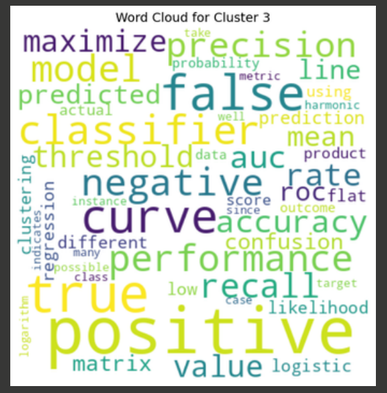


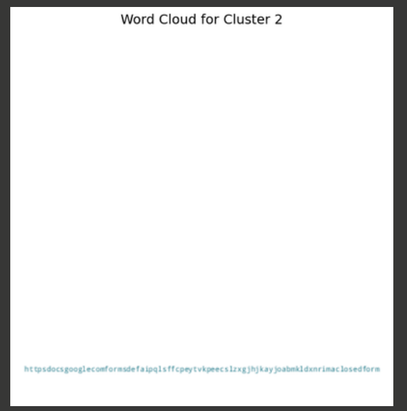
Each Session Word Cloud

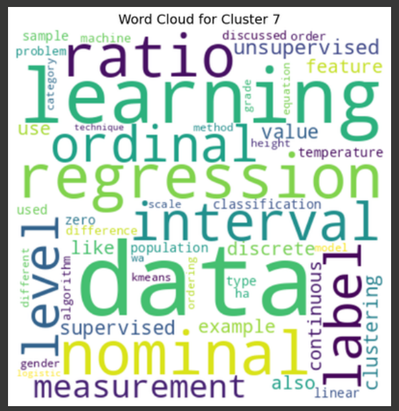






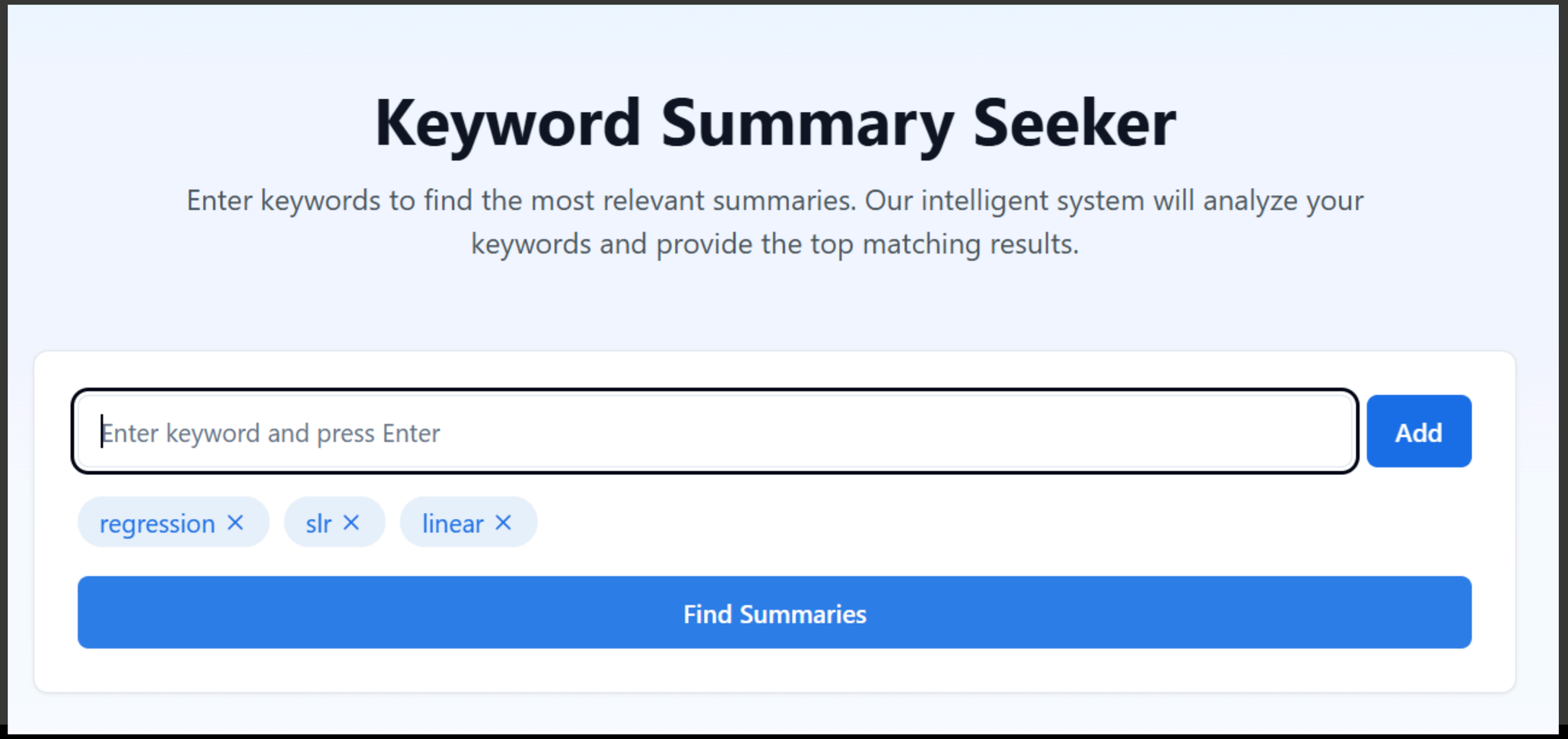






As it was google drive link in data

User-Friendly App Creation



What functionalities does the app offer?

Keyword-Based Search Input

- Users can enter any number of keywords related to a topic of interest

Automated Topic Relevance Detection

- The app feeds the keywords into a Python backend which analyzes session content for relevance

Smart Summary Selection

- The Python code identifies and returns the top 3 most relevant summaries based on the keywords provided

Minimalistic Output Display

- The selected summaries are shown in a clean, separate text window for easy reading—keeping the interface simple and focused.

What all we learnt

- Mastered full NLP pipeline: preprocessing, tokenization, embedding, clustering, ranking, and visualization.
- Understood the tradeoffs between frequency-based methods (TF-IDF) and semantic methods (Word2Vec).
- Designed a hybrid embedding model (TF-IDF × Word2Vec) that performed better than standalone methods.
- Learned the strengths and limitations of metrics like Silhouette Score, and why manual validation is crucial.
- Built an end-to-end unsupervised system to re-associate session summaries without any labels.
- Developed a session-wise summary retrieval app with keyword-based search, ranking, and clustering visualizations.

Hurdles Faced

- Summaries were inconsistent — different writing styles, tones, and levels of detail made preprocessing difficult.
- Many students used general vocabulary, while some used sir's phrasing — complicating semantic grouping.
- Traditional clustering with Bag of Words & Euclidean distance failed to capture meaning — led to mixed or meaningless clusters.
- Silhouette Score often gave misleading results — high scores despite incorrect clustering due to vocabulary overlap.
- Deep learning methods like BERT were impractical — too resource-heavy and prone to bias from writing style or grammar.
- Needed to balance multiple featurization techniques and design a ranking logic that made sense both mathematically and contextually

Conclusion

- **Unsupervised learning is powerful when labels are missing — but only when paired with semantic reasoning.**
- **Our pipeline went from raw, noisy summaries to structured clusters, ranked insights, and searchable outputs.**
- **Word clouds helped surface key themes; summary ranking helped in relevance-based retrieval.**
- **Manual validation showed that our hybrid model closely matched how real students think and summarize.**
- **The final app makes navigating and exploring the summaries intuitive, personalized, and highly informative.**
- **In essence: the project taught us how to bring structure to unstructured data — not just technically, but interpretively.**

Thank You :)