StackExchange Prediction and Analysis

Project for Advanced Data Mining Course

Tomasz Kawiak Tomasz Makowski Mateusz Mazur

December 19, 2025

Contents

Introduction

This project investigates how deep learning and embedding methods can be applied in analysis and prediction of properties derived from StackExchange (StackOverflow) questions. Using the StackExchenge API, we collected a dataset of 100,000 questions together with their metadata, textual context, and tag information. The analysis focuses on transforming the high-dimensional, sparse and highly imbalance data into representation suitable for learning.

A major technical challenge addressed was managing the extreme cardinality and high dimensionality of the data: specifically, compressing over 22,000 unique question tags into a meaningful, lower-dimensional space, and then leveraging 4096-dimensional embeddings for both multi-label classification (tag prediction) and high-variance regression (question score prediction). To accomplish this, we embedded all tags, titles and question bodies using the `qwen3-embedding:8b` model. For tags, we have applied a series of dimensionality reduction and clustering strategies. After evaluating UMAP (McInnes, Healy, and Melville 2020) + HDBSCAN (McInnes, Healy, and Astels 2017), Birch (Zhang, Ramakrishnan, and Livny 1997), and agglomerative approaches, we adopted **Recursive Spherical K-Means**, which produced 100 tag centroids that preserve coverage of all original tags.

The core objectives pursued throughout this project were:

1. **Data Exploration and Feature Engineering (EDA):** To quantify the sparsity and distribution characteristics of key features such as tag frequency, temporal metrics related to answer acceptance, and question scoring.
2. **Dimensionality Reduction:** To overcome computational limits and improve model tractability by intelligently reducing the space of 4096-dimensional embeddings and the semantic space of 22,753 unique tags.
3. **Tag Classification:** To design and evaluate neural network architectures capable of predicting question categories based on textual input from the title and body embeddings.
4. **Score Regression:** To assess the intrinsic predictability of question quality (measured by score) directly from the learned semantic embeddings.

The methodology relied heavily on the `qwen3-embedding:8b` model for vector representation and incorporates robust machine learning techniques such as **Recursive Spherical K-Means** for unsupervised clustering, and **Asymmetric Loss (ASL)** and **Cross-Attention** mechanisms for enhanced deep learning performance.

The Data

The analysis is based on a dataset comprising of **100,000 questions** extracted from the StackExchange platform (StackOverflow) using StackExchange API. The aim was to get to know the data characteristics and see what challenges may arise during modeling as well as what in particular can be predicted from the textual content.

Data Characteristics and Preprocessing

The raw dataset contained eleven distinct features covering textual content, metrics, and acceptance status:

#	Column	Type
0	`title`	`string`
1	`has_accepted_answer`	`bool`
2	`accepted_answer_score`	`float64`
3	`time_to_accepted_answer_hours`	`float64`
4	`question_score`	`int64`
5	`question_text`	`string`
6	`num_tags`	`int64`
7	`tags`	`string[]`
8	`accepted_answer_id`	`float64`
9	`accepted_answer_length_chars`	`float64`
10	`accepted_answer_length_tokens`	`float64`

Initial data hygiene involved dropping 8 duplicate questions identified by their ID.

Analysis of Answer Metrics and Sparsity

A crucial finding from the exploratory data analysis (EDA) was the significant sparsity in the answer-related features:

- A total of 39,938 questions had an accepted answer, while 60,054 did not.
- However, only **12,000** of these accepted answers had non-null values for temporal metrics.

Analysis of the `time_to_accepted_answer_hours` for this small subset revealed that the distribution, when log-transformed, exhibited a multimodal structure. This suggests that answers are accepted across different temporal regimes, possibly corresponding to simple versus complex problems, or different subject areas (fig. 1).

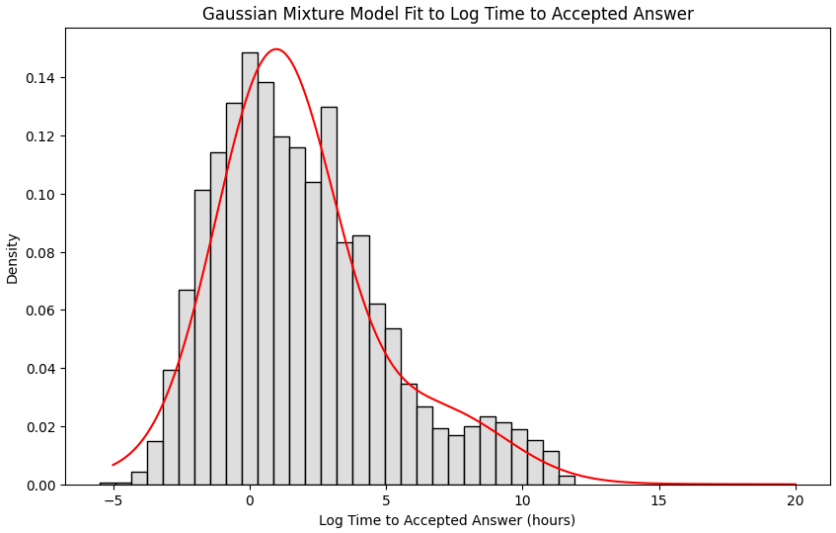

Figure 1: Gaussian Mixture Model Fit to Log Time to Accepted Answer

This result led to the conclusion that answer-based analysis was challenging due to the lack of sufficient data points with defined accepted answer characteristics. Scraping more data, just to analyze only a fraction of questions seemed infeasible, that's why we dropped this idea.

Tag Cardinality and Filtering

The raw dataset contained an excessively large vocabulary of **22,753 unique tags**. The top five most frequent initial tags were `python` (1528), `c#` (746), `javascript` (703), `c++` (689), and `java` (592). For simplicity sake, we initially focused on the 7,684 most frequent tags. The majority of questions in this subset had only one tag (38,547), although multi-label instances were present (3,015 questions had 2 tags; 464 had 3).

To create a manageable feature space for initial classification attempts, two approaches were considered:

1. **Frequency-Based Filtering:** Limiting the analysis to the most popular programming languages (e.g., Python, C#, Java). This approach resulted in a subset of 42,037 questions tagged with a set of 9 popular programming languages.
2. **Semantic Clustering:** Using high-dimensional embeddings and clustering algorithms to group semantically similar tags.

After a preliminary analysis, the second approach was chosen to retain semantic richness while reducing dimensionality and making the challenge more fun and interesting.

Question Score Distribution

The raw scores (`question_score`) ranged widely from -20 up to 27,487. The score distribution was heavily skewed around zero (fig. 3). We considered grouping this continuous variable into five classes of uneven frequencies:

- **Bad:** $(-\text{inf}, -1]$

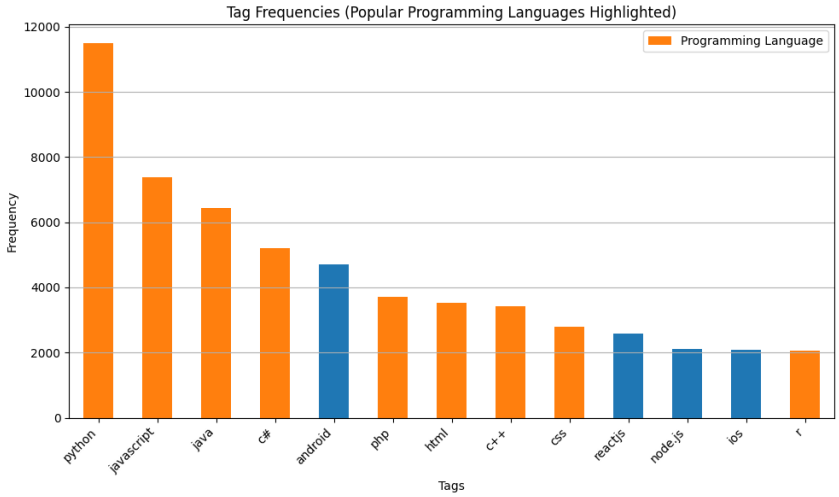

Figure 2: Tag frequencies for most frequent tags

- **Neutral:** 0
- **Good:** $(1, 3]$
- **Very Good:** $(3, 20]$
- **Excellent:** $(20, \text{inf})$

But ultimately, the decision was made to treat score prediction as a regression problem to preserve the granularity of the data.

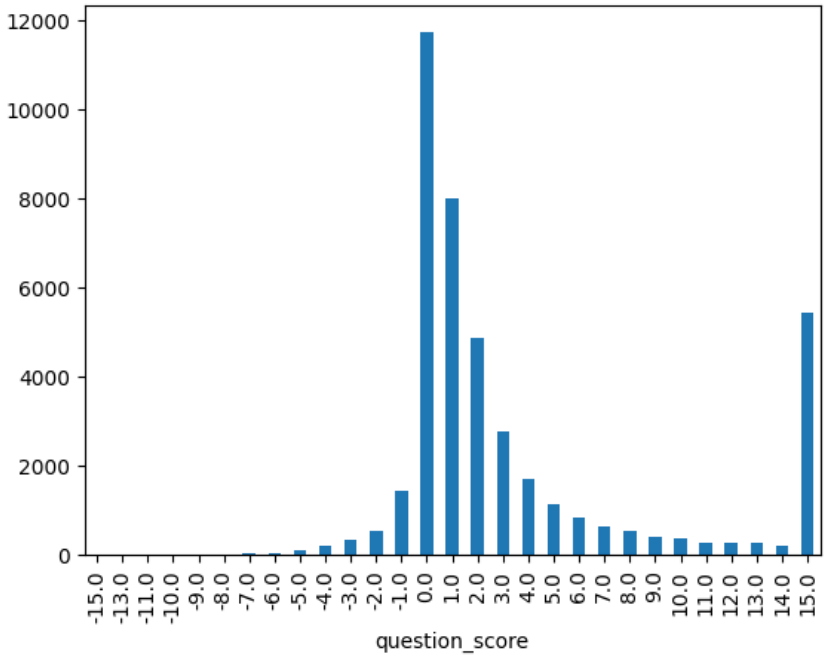

Figure 3: Question Score Distribution

Data gathering summary

Finally, we retained the following columns for further analysis:

- `title` (textual content)
- `question_text` (textual content)

- `tags` (target for classification)
- `question_score` (target for regression)

The final version of the dataset contained **99 992 unique questions**.

The Methodologies

The path to generating robust predictive models required significant methodological investment, particularly in handling the immense size and complexity of the embedded text data.

Embedding Generation

To transform the textual data (titles, tags, and question bodies) into numerical feature representations, we utilized `qwen3-embedding:8b`, an open-source model capable of generating **4096-dimensional vectors** (Qwen Team 2025). Textual data (titles and question bodies) was transformed into **4096-dimensional vectors** using the open-source embedding model `qwen3-embedding:8b`.

We implemented two distinct embedding strategies:

1. Global Document Embedding (Baseline)

- This approach served as the baseline due to its simplicity and relative computational speed.
- Embed each question body, title and tag individually into an embedding vector.
- This method assumes that the semantic context of the entire document can be effectively compressed into a single 4096-dimensional vector (using `float64` precision) without significant information loss.
- Computation required approximately 6 hours using the `ollama` library (Ollama 2025). The resulting dataset occupied 3.3 GB of storage.

2. Sequential Token Embedding

- To address potential information loss in the baseline approach, we hypothesized that a single vector might fail to capture complex dependencies in longer texts.
- Instead of pooling the text into one vector, we maintained a sequence of embeddings to preserve token-level knowledge. We defined a fixed sequence length of 4 tokens for the title and 32 tokens for the body, resulting in a distinct embedding vector for each token.
- Implementation changes:
  - Due to the limitations of `ollama` in the terms of appropriate tokenizer for our model, as well as inefficient resource management during embedding, we've switched to *Hugging Face* `transformers` library.
  - Because of the exponential increase in data size and compute time, we reduced the floating-point precision from `float64` → `float32`.
- These optimizations reduced the estimated compute time from 40 hours to approximately 13 hours.
- The resulting embeddings required 27 GB of space. Due to memory constraints preventing the dataset from being loaded entirely into RAM, we utilized the Hierarchical Data Format (HDF5) for efficient storage and access (The HDF Group 1997-2025).

Embedding Analysis

After the embedding phase, we wanted to verify the validy of the topology and density of the resulting embedding space. Specifically, we wanted to verify that the embeddings captured sufficient semantic overlap between questions to facilitate meaningful clustering.

To measure this, we analyzed the *Nearest-Neighbour Cosine Similarity*. Using a subset of the question embeddings, we performed the following steps:

- Normalized the embeddings (the resulting embeddings from `ollama` theoretically should be normalized, but it's better to normalize it anyway, since it does not affect the data).
- We utilized the `NearestNeighbors` algorithm (from `scikit-learn`) to locate the closest non-identical neighbor ($k = 1$) for each data point.
- We calculated the cosine similarity for these pairs, defined as $1 - \text{cosine\_distance}$.

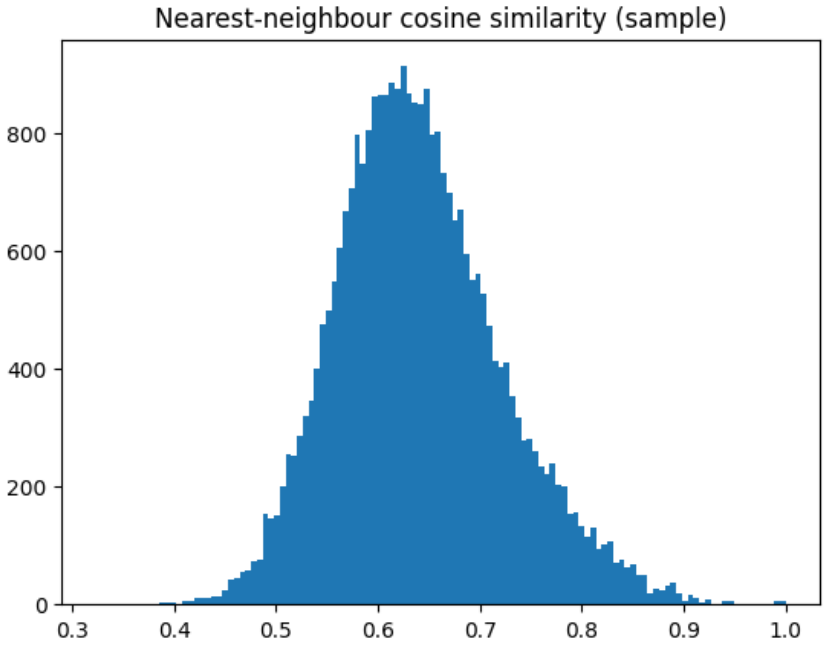

Figure 4: Nearest-Neighbour cosine similarity distribution

As we can see in Figure fig. 4, the distribution of nearest-neighbor similarities is approximately Gaussian and centered around 0.65.

- The lack of data points near 0.0 indicates that very few questions are "isolated" in the vector space; almost every question has a semantically related counterpart.
- The unimodal distribution suggests a well-structured manifold where local neighborhoods are consistent. This confirms that the embedding model (`qwen3-embedding`) successfully mapped semantically similar questions to adjacent regions in the high-dimensional space, providing a strong foundation for the subsequent clustering phases.

Dimensionality Reduction and Clustering

The high dimensionality (4096 dimensions) and high cardinality (22,753 unique tags) presented immediate barriers to traditional clustering techniques, not to mention classification task. As we still wanted to retain semantic richness in the tag space, we explored various dimensionality reduction and clustering strategies.

Initial tries (HDBSCAN and UMAP)

1. **HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise)** was initially rejected because memory requirements exceeded the available 64GB of RAM.
2. **UMAP (Uniform Manifold Approximation and Projection)** was adopted as a non-linear dimension reduction technique (e.g., reducing to 5 components), coupled with HDBSCAN clustering. Optimization of UMAP and HDBSCAN parameters was attempted using **Optuna**, guided by unsupervised metrics like the Caliński-Harabasz index, Silhouette score, and Cluster persistence.

This combined approach failed to yield coherent results. The clusters were highly imbalanced, with large numbers of points assigned to a single noise cluster or small, trivial clusters (visible in fig. 5 and fig. 6). This failure was theorized to stem from the violation of UMAP's underlying assumption that data should be uniformly distributed on a Riemannian manifold, as dense embeddings often exhibit varying local geometry.

Recursive Spherical K-Means Clustering

To achieve a stable reduction of the tag space to **100 semantic centroids**, a **Recursive Embedding and Clustering (REAC)** approach, inspired by industry techniques, was implemented.

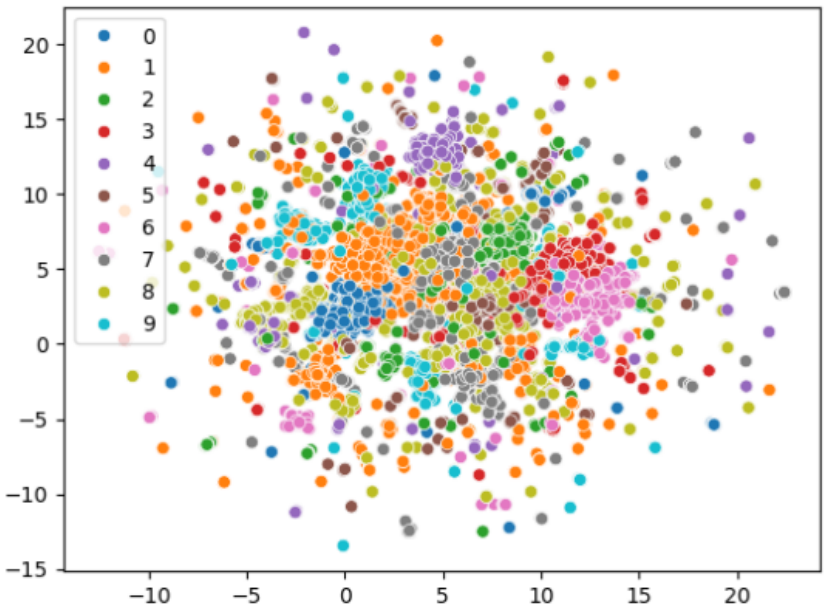

Figure 5: Naiive UMAP reducing tag embeddings to 2 components

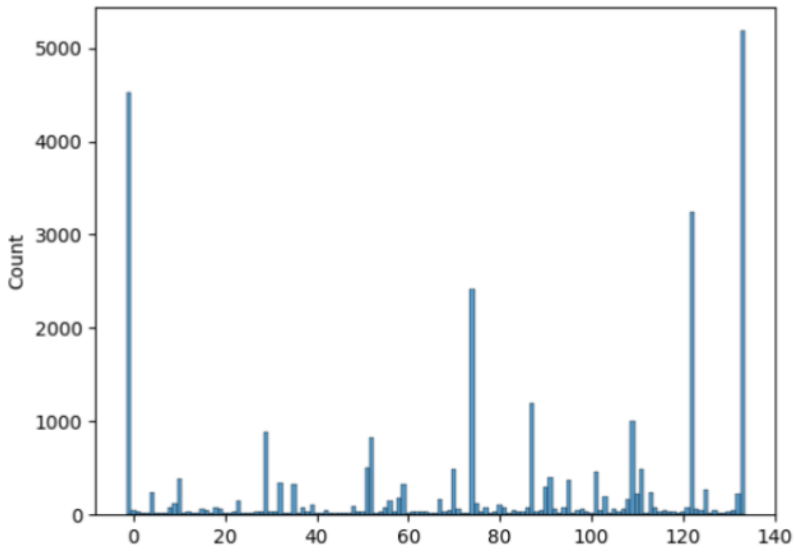

Figure 6: Histogram of labels resulting from UMAP reduction coupled with HDBSCAN

1. **Tag Filtration:** Tags with a low frequency (threshold < 100) were initially removed, resulting in a dataset reduced to **411 unique tags** across 90,205 records.
2. **Clustering: Recursive Spherical K-Means** was employed on these 411 tags. Spherical K-Means is particularly advantageous for textual embeddings as it optimizes distance based on **cosine similarity**. This process successfully yielded the target 100 centroids. Unlike in the case with HDBSCAN, this time the resulting clusters showed a stable distribution of tags per centroid fig. 7 and focused on general ideas connecting the grouped tags, e.g.: the cluster that could have been described as "databases" included: `database`, `mongodb`, `sql`, `postgresql`, `mysql`, etc..
3. **Orphan Assignment:** The 9,787 infrequent "orphan tags" were assigned to their nearest centroid based on cosine similarity, ensuring that all 22,753 original tags were mapped to the final 100 classes.

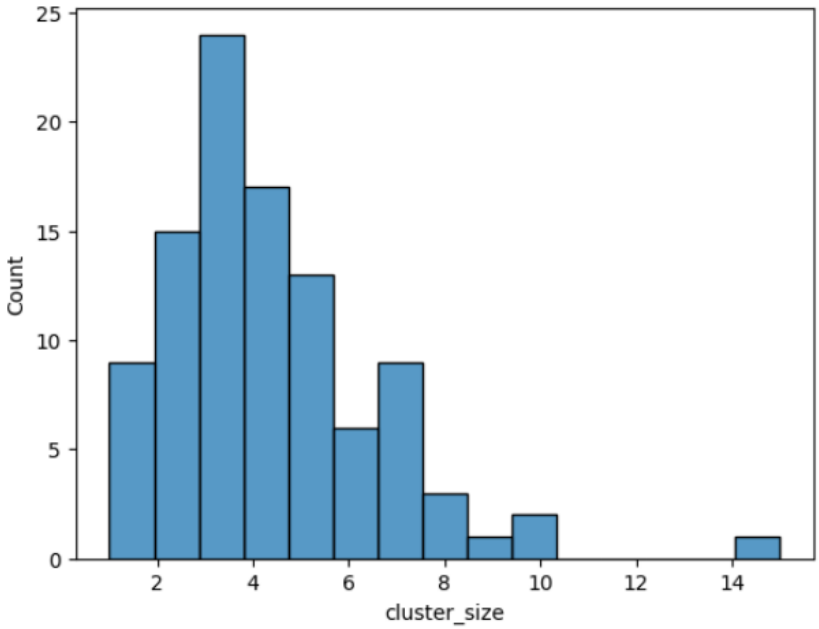

Figure 7: Distribution of Sizes of Clusters found using Recursive Spherical K-Means

The Experiments

Tag Prediction

The primary classification goal was to predict the appropriate semantic tag centroid(s) for a given question using its 4096-dimensional title and body embeddings.

XGBoost Baseline

As a baseline, an XGBoost model was trained on question body embeddings. To simplify the initial evaluation, the inherently multi-label task was reduced to a multiclass classification problem. The target for each question was mapped to a single label via a majority vote mechanism, selecting the centroid (or tag group) containing the highest frequency of the question's original tags.

This XGBoost model trained for 599 minutes. It achieved a **Weighted F1 Score of 0.6882** (and accuracy of 0.6928). The limitation of this approach was its inability to effectively model the complex, non-linear semantic interactions embedded in the 4096-dimensional vectors, forcing it to treat the dimensions largely independently. Nevertheless, this result provided a solid benchmark for subsequent deep learning models.

Neural Network Architectures

Neural Networks (NNs) were adopted to natively support multi-label output and leverage advanced techniques like Attention Mechanisms.

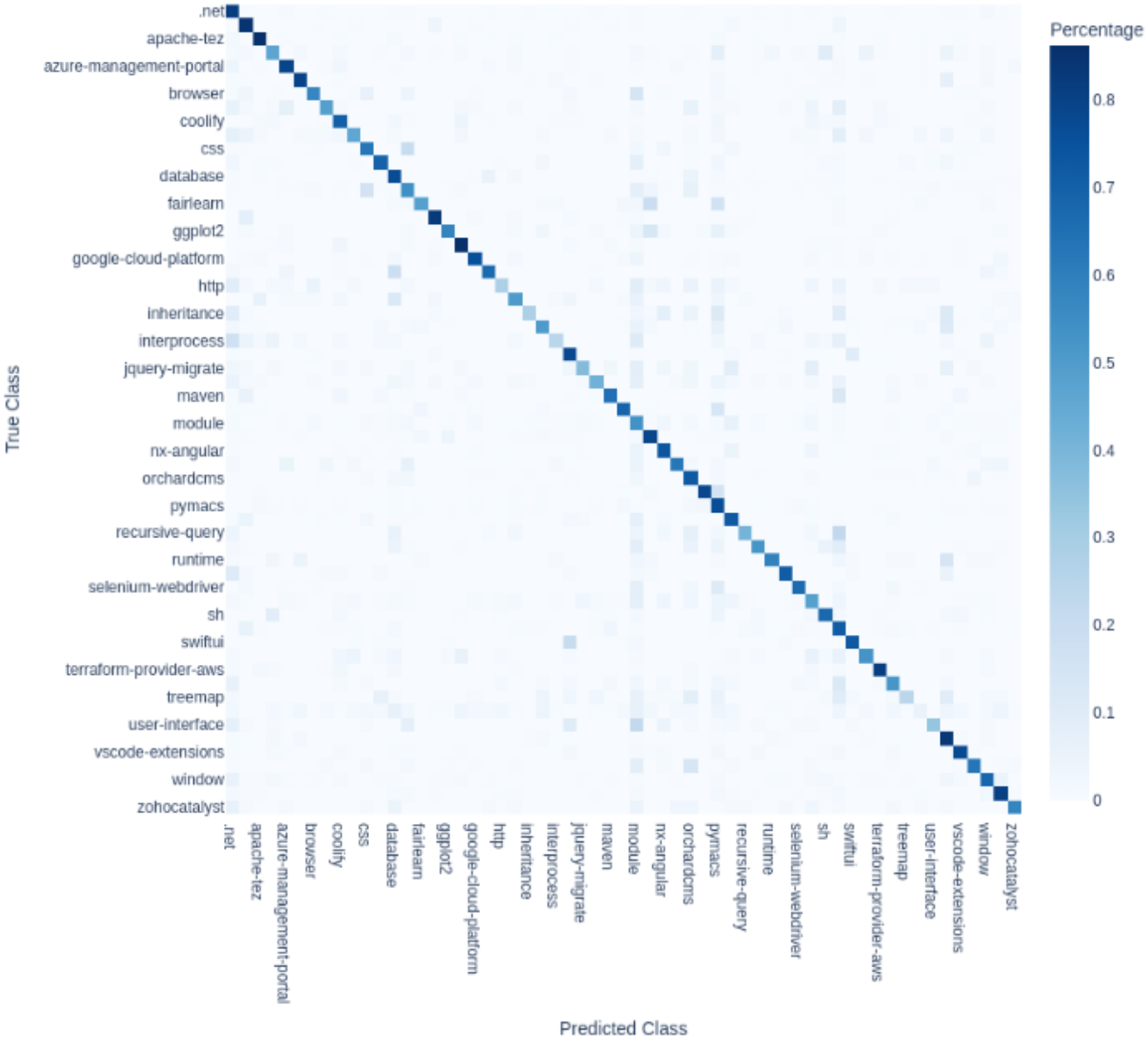

Figure 8: Confusion Matrix of XGBoost Classifier

Simple MLP Baseline A basic Multi-Layer Perceptron (MLP) trained on the question body embeddings with `BCEWithLogitsLoss` achieved a weighted F1 score of approximately 0.6790. This confirmed the need for richer architectures that could utilize both title and body context simultaneously.

Dual-Stream Fusion Network (DSF) The DSF models used dual streams to process Title embeddings and Body embeddings separately before combining them.

1. **DSF with Multi-Head Self-Attention (MHSA):**
   - This initial fusion mechanism involved concatenating the processed embeddings and applying MHSA, inspired by multi-modal classification work.
   - Initial training suffered from rapid overfitting. This was mitigated by applying a higher Dropout rate (0.5), switching to the AdamW optimizer (decoupling weight decay for better generalization), and adopting **Asymmetric Loss (ASL)**.
   - **ASL** was critical for addressing the class imbalance inherent in multi-label classification (where most labels are negative for any sample). ASL uses focusing parameters ($\gamma_- = 4, \gamma_+ = 1$) to aggressively down-weight easy negative examples, forcing the model to focus on the difficult ones and on positive examples.
   - Result: **F1 Weighted 0.7110**.
2. **DSF with Cross-Attention Fusion (Optimal Model):**
   - To further combat overfitting and improve feature interaction, the MHSA was replaced with **Cross-Attention**, allowing the typically concise **Title embedding to "query" the verbose Body embedding** to extract relevant features.
   - Additional regularization, **Manifold Mixup**, was applied to the embeddings during training to encourage smoother decision boundaries in the latent space.
   - Trained over 100 epochs (~60 minutes) using OneCycleLR scheduling, this model achieved the best outcome: **F1 Micro 0.7253** and **F1 Weighted 0.7196**.

Table 2: Summary of Tag Prediction Results (Weighted F1 Score)

Model	F1 Score (Weighted)
XGBoost (Multiclass Approximation)	0.6882
Baseline MLP	0.6790
DSF with MHSA Fusion	0.7110
DSF with Cross-Attention Fusion	**0.7196**

Attention Analysis Visualization

Post-training analysis included visualizing the internal workings of the Cross-Attention mechanism to understand how the 8 attention heads weighted the relationship between the Title and Body embeddings for specific samples. This visualization provided diagnostic insight into the model's decision process (Figures 9 and 10).

Figure 9: Cross-Attention Alignment per Head (Sample #0)	**Figure 10: Cross-Attention Alignment per Head (Sample #2)**
(**TODO?**): Insert Figure: Cross-Attention Alignment per Head (Sample #0) (Source:)	(**TODO?**): Insert Figure: Cross-Attention Alignment per Head (Sample #2) (Source:)

Score Prediction

The goal was to predict the raw integer question score using only the embedded textual content (regression task). This task was inherently challenging due to the high variance and sparse nature of scores (mean score of 23.55, but max score of 27,487, with most scores clustered near zero).

Baseline Comparison (Traditional ML vs. Embeddings)

An initial exploration using traditional feature extraction (TF-IDF) and statistical dimensionality reduction (Truncated SVD, a form of Latent Semantic Analysis) provided a challenging context for the deep learning models: * The mean baseline achieved an R^2 of -0.000196 (Test RMSE 149.97). * The best traditional model, **TF-IDF + SVD + Ridge**,

managed a Test R^2 of **0.0074** (Test RMSE 275.05) on the raw score target. This low R^2 score reinforced the difficulty of explaining score variance using extracted features alone.

Deep Learning Regressors

Single-stream MLPs (using only Title or Body embeddings) and the dual-stream DSF architectures (MHSA and Cross-Attention variants) were adapted for regression using Mean Squared Error (MSE) loss. The input features included the embeddings and a normalized tag count feature (clipped to).

All deep learning models significantly outperformed the traditional mean baseline ($R^2 \approx 0.000$).

Table 3: Score Regression Performance on Test Set (using Embeddings)

Model	Input	Test RMSE	Test R^2
DSF-MHSA Regressor	Title + Body Embeddings	134.24	**0.199**
DSF-CrossAttention Regressor	Title + Body Embeddings	135.27	0.186
Body MLP	Body Embedding Only	137.57	0.158
Title MLP	Title Embedding Only	138.73	0.144
Mean Baseline	Constant	149.97	0.000

The **DSF-MHSA Regressor** achieved the highest performance, explaining approximately **19.9%** of the score variance on the test set. Diagnostic analysis showed that predictions tended to cluster accurately near zero/low scores, but the model struggled significantly to predict high scores correctly (Figure 11).

Figure 11: Predicted vs Actual Scores (DSF-MHSA, Filtered Range)

(**TODO?**): Insert Figure: Predicted vs Actual Scores (DSF-MHSA, Filtered to <200) (Source: or)

Figure 12: Residual Distribution (DSF-MHSA, Filtered Range)

(**TODO?**): Insert Figure: Residual distribution (DSF-MHSA, Filtered to <200) (Source: or)

Conclusions and Future Work

The overall project successfully navigated several data processing and modeling complexities inherent in large-scale textual data.

Key Successes and Insights

- **Robust Tag Reduction:** The greatest technical victory in the data preparation phase was the establishment of the **Recursive Spherical K-Means** pipeline, effectively reducing the 22,753 original tags to 100 semantically coherent centroids. This step was vital for making multi-label classification computationally feasible after the failure of unsupervised methods like UMAP/HDBSCAN optimization.
- **Optimal Classifier Architecture:** The use of **Dual-Stream Fusion Networks** demonstrated a clear advantage over both traditional ML (XGBoost) and simple MLPs for multi-label classification. The best results were achieved by the **DSF with Cross-Attention Fusion (F1 Weighted 0.7196)**, incorporating both **Asymmetric Loss** to manage label imbalance and **Manifold Mixup** to improve generalization.
- **Score Prediction Difficulty:** The consistently low R^2 values across all regression experiments confirm that predicting Stack Overflow scores from purely semantic content embeddings is inherently difficult due to the social/external factors influencing a question's eventual popularity (score).

Summary of Best Results

Task	Best Model	Key Metric	Result
Tag Prediction (Classification)	**DSF Cross-Attention**	F1 Weighted	**0.7196**
Score Prediction (Regression)	**DSF-MHSA Regressor**	Test R^2	**0.199**

Future Work

Future research should focus on mitigating the score prediction problem by incorporating features that capture non-textual quality signals, such as user reputation or time-of-day posting biases, which were outside the scope of this content-based embedding analysis. For classification, exploring even deeper transformer architectures for sequence-aware embedding generation (if computational resources allow) could further enhance performance beyond the current pooled embedding methods.

McInnes, Leland, John Healy, and Steve Astels. 2017. "Hdbscan: Hierarchical Density Based Clustering." *The Journal of Open Source Software* 2 (March). https://doi.org/10.21105/joss.00205.

McInnes, Leland, John Healy, and James Melville. 2020. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." https://arxiv.org/abs/1802.03426.

Ollama. 2025. "Ollama: Get up and Running with Llama 3.2, Mistral, Gemma 2, and Other Large Language Models." https://github.com/ollama/ollama.

Qwen Team. 2025. "Qwen3-Embedding-8B: A Versatile Text Embedding Model." https://huggingface.co/Qwen/Qwen3-Embedding-8B; Hugging Face.

The HDF Group. 1997-2025. "Hierarchical Data Format 5 (HDF5)." https://www.hdfgroup.org/HDF5/.

Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. 1997. "BIRCH: A New Data Clustering Algorithm and Its Applications." *Data Mining and Knowledge Discovery* 1 (2): 141–82. https://doi.org/10.1023/A:1009783824328.