StackExchange Prediction and Analysis

Project for Advanced Data Mining Course

Tomasz Kawiak   Tomasz Makowski   Mateusz Mazur

December 20, 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 (T. 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`

#	Column	Type
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).

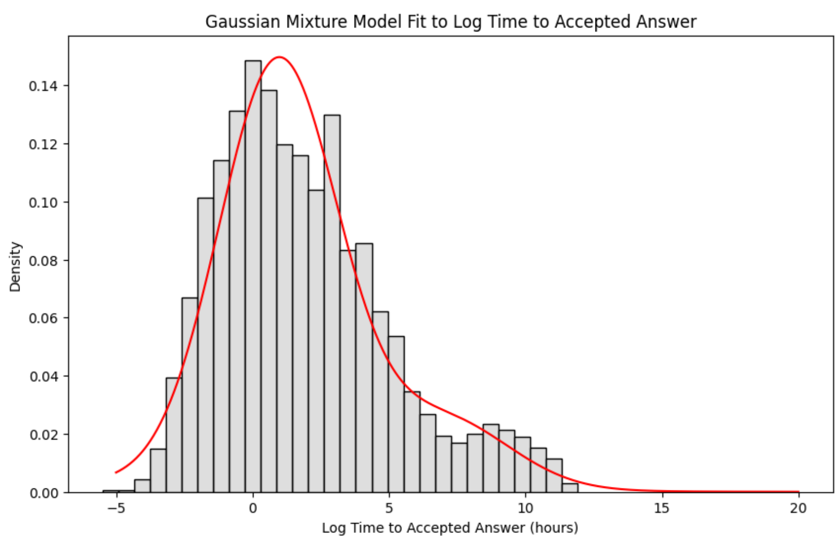

Figure 1: Gaussian Mixture Model Fit to Log Time to Accepted Answer

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.

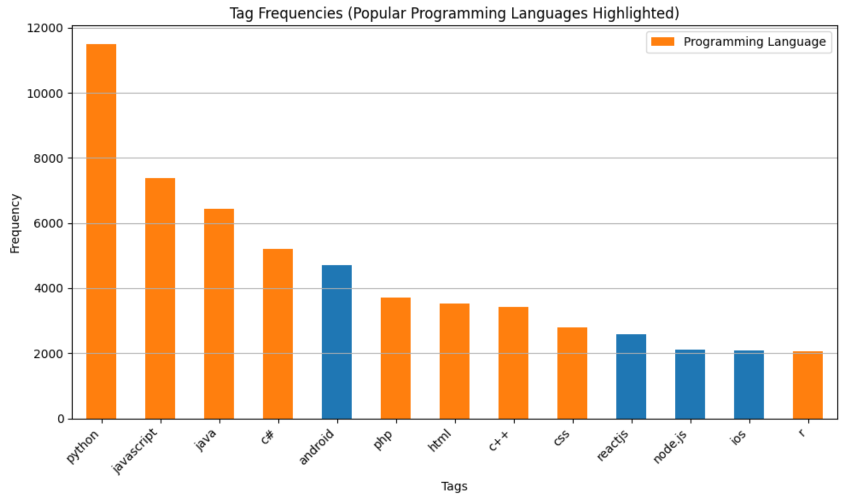

Figure 2: Tag frequencies for most frequent tags

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

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

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`**.

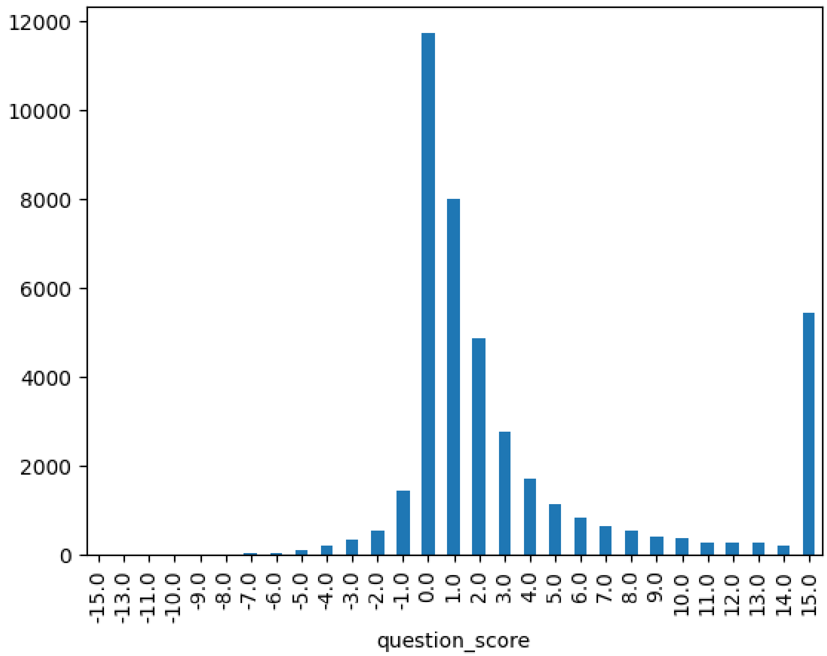

Figure 3: Question Score Distribution

We implemented two distinct embedding strategies:

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.

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 validity 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 $1 - \text{cosine\_distance}$.

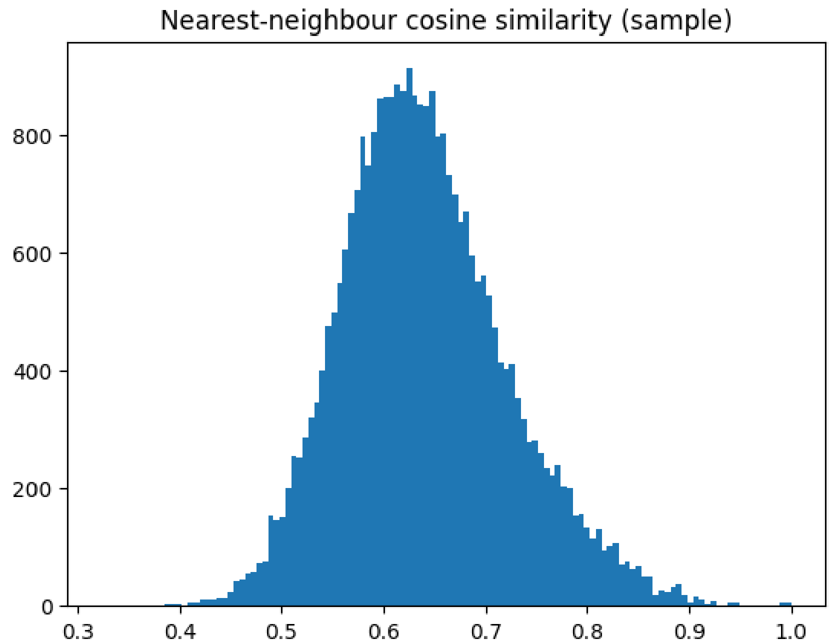

Figure 4: Nearest-Neighbour cosine similarity distribution

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.

Tag Dimensionality Reduction and Clustering

The tag space presents two major challenges: high dimensionality (4096-dimensional embeddings) and high cardinality (22,753 unique tags). These properties make traditional clustering methods difficult to apply directly. Because we aimed to preserve as much semantic structure as possible, we investigated several dimensionality reduction and clustering strategies.

HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) (McInnes, Healy, and Astels 2017) was initially considered, due to its ability to identify clusters of varying density and to naturally model noise. However, the method proved infeasible at our scale. During graph construction, the algorithm attempted to allocate a dense distance matrix, resulting in more than 64 GB of RAM usage:

```
MemoryError: Unable to allocate 74.5 GiB for an array with shape (99992, 99992) and data type
↪  float64
```

Even incorporating Birch pre-clustering (T. Zhang, Ramakrishnan, and Livny 1997), which is often recommended to reduce memory footprint, did not sufficiently mitigate these requirements.

UMAP

Given its popularity for high-dimensional manifold learning and its strong community reports, particularly its successful use in document embedding tasks such as the 20 Newsgroups dataset, we next explored UMAP (Uniform Manifold Approximation and Projection) (McInnes, Healy, and Melville 2020) as a potential solution. Encouraged by these findings, following community best practices, we tested two approaches:

- Single stage with 'pure' UMAP
- Two-stage with PCA → UMAP

The rationale for the second approach comes from the UMAP's reliance on pairwise distances when constructing its topological graph. In very high-dimensional spaces, distance measures tend to concentrate, causing points to appear nearly equidistant. Applying PCA first captures the dominant variance structure and reduces sparsity effects before manifold learning.

For the single stage approach, we applied UMAP directly to the 4096-dimensional vectors using the cosine metric (appropriate given that the underlying embeddings were already normalized thanks to `ollama`). Reducing directly to two dimensions produced the projection shown in fig. 5, which exhibited significant overlap and very poor global structure.

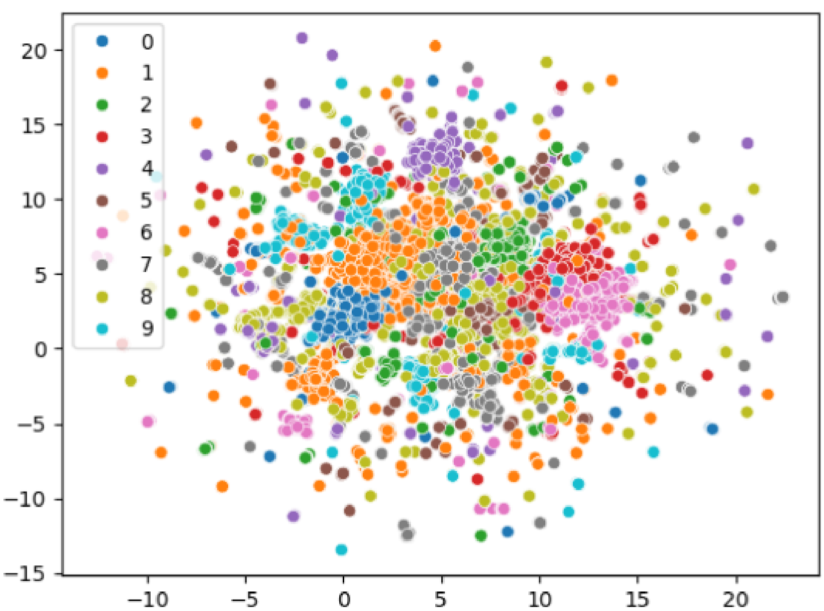

Figure 5: Naiive UMAP reducing tag embeddings to 2 components (colors are just for aesthetic reasons).

After brief hyperparameter search using optuna, we've decided to scrap the single-stage approach, in favor of the two-stage one.

As the first stage, PCA was picked as the best candidate, due to it's variance maximization, relative simplicity, and little computation overhead.

Optuna hyperparameter search

To improve UMAP's performance, we conducted a hyperparameter search using Optuna (Akiba et al. 2019). For optimization, a suitable objective function was required. We first experimented with the Calinski–Harabasz index (Caliński and Harabasz 1974), but due to its tendency to assign inflated scores to degenerate solutions (e.g., one or two dense clusters), we replaced it with a weighted combination of:

- Silhouette Score
- Calinski-Harabasz score
- Custom penalty for generating clusters with very few points.

We tested both a flat optimization procedure and a nested one. In the nested variant, the outer loop optimized PCA hyperparameters, and the inner loop optimized UMAP parameters conditioned on the PCA output.

Neither approach produced coherent clustering. The resulting label distributions (fig. 6) were dominated by either a large noise cluster or numerous trivial micro-clusters. This instability pointed to deeper issues in the reduction pipeline:

- The evaluation metrics themselves were insufficiently sensitive to the structure we hoped to recover.
- Available compute limited the ability to explore more.
- Most importantly, the underlying assumptions of UMAP were violated:
  - UMAP assumes the data is sampled from a uniformly distributed Riemannian manifold, but dense embedding spaces often exhibit highly non-uniform local geometry.

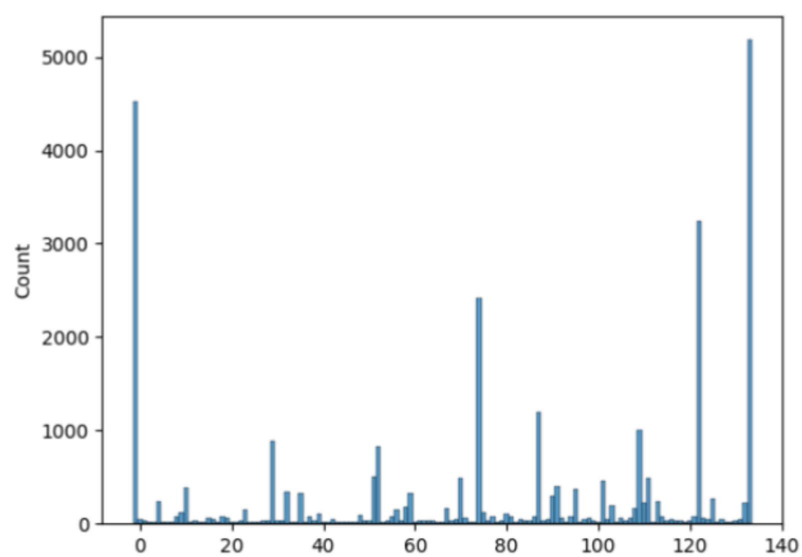

Figure 6: Histogram of labels resulting from UMAP reduction coupled with HDBSCAN. Each y value corresponds to a single cluster.

Variance Loss in PCA

A more fundamental limitation emerged when examining the PCA stage itself. To measure how much information a PCA stage would discard, we computed the explained variance on a representative sample of 10,000 embeddings. The results revealed that the embedding space is extremely flat: the first principal component alone explains a negligible fraction of the variance, and adding more components yields slow, sublinear improvement. Even with the top 50 PCs, only 38% of the total variance is recovered. This behavior is illustrated in fig. 7.

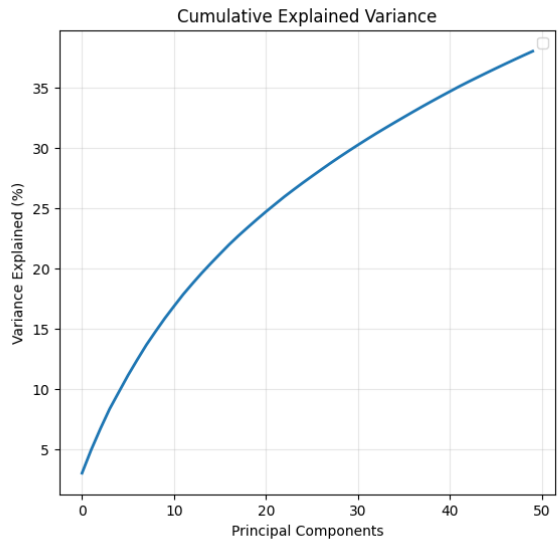

Figure 7: Cumulative variance curve of PCA.

This structure is also visible when comparing the distribution of a raw embedding dimension to the distribution of the first principal component. As shown in fig. 8, a random raw dimension is tightly concentrated around zero, indicating very low variance. In contrast, PC_1 has a much broader distribution, reflecting the fact that most of the meaningful variance in the data collapses.

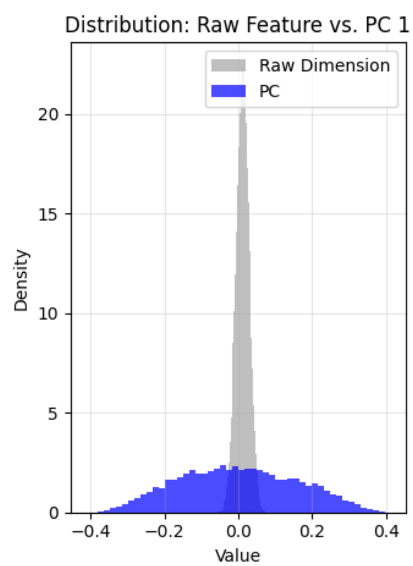

Figure 8: Cumulative variance curve of PCA.

To reiterate, metric limitations, computational constraints, and mismatches between the UMAP's assumptions and the geometry of dense embedding space, combined with the extensive optuna hyperparameter search, for which the best candidate identified by the optimization process, the PCA alone discarded approximately 62% of the variance present, motivated us to search for different approach to dimensionality reduction.

Recursive K-Means Clustering

Our initial approach was inspired by methodologies for hierarchical data organization, such as those described in engineering blogs by Spotify dealing with large-scale recommendation systems. We designed a Recursive K-Means Clustering algorithm to structure the tags into a navigable tree.

The core concept involves recursively partitioning the tag embedding space. Starting with the root node containing all tags, we apply K-Means to split the data into k clusters. This process is repeated for each resulting cluster until a maximum depth is reached or a cluster becomes too small to split further. For this algotithm, we've relied on Euclidean distance.

The big difference from the previous algorithms is that we do not reduce the data itself, but rather **find the best representation for the predefined amount of classes**.

Table 2: Sample results from initial K-Means Clustering run.

Tag	closest_dist	cluster_mean	tag_embedding	cluster_size	top_5_popular_tags
facebook	0.089584	[0.004879954, ...]	[0.002688237, ...]	94	[instagram-api, instagram, ...]
integer	0.040206	[0.027635492, ...]	[0.03159611, ...]	370	[intervals, differential-equations, ...]
storage	0.105308	[0.008282, ...]	[0.020607475, ...]	134	[dicomweb, remotestorage, ...]
tf-idf	0.070302	[0.027263727, ...]	[0.024365697, ...]	243	[wpml, locale, ...]
qt	0.074746	[0.01614129, ...]	[0.016203284, ...]	106	[qt, qml, pyqt, qasync, qt6]

Recursive Spherical K-Means Clustering

While effective for spatial data, Euclidean distance is often suboptimal for high-dimensional semantic embeddings, where the direction of the vector typically encodes more semantic meaning than its magnitude. That's why, to better capture the semantic relationships between tags, we evolved our approach to Recursive Spherical K-Means Clustering (Hornik et al. 2012). This variation partitions the data by minimizing the cosine dissimilarity rather than the Euclidean distance, effectively clustering data points on the surface of a hypersphere (hence, the spherical in Spherical K-Means).

Implementation Details

After investigating the tag frequency in our dataset, we've decided to impute all tags, which occured less frequent than 100 times. This resulted in 411 unique tags and in 90205 questions whith tag. This operation did not changed the overall characteristic of the dataset as we can see in the fig. 9

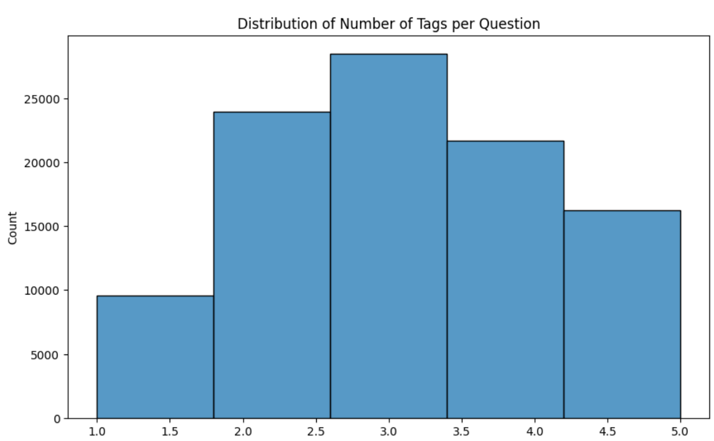

Figure 9: Distribution of number of tags per question

With the orphan quesions (the ones without tags), we assigned them to the nearest embedding based on the cosine similarity of their tag.

For example:

```
closest_centroid_tag(tags,centroid_tags, 'ntp')
('datetime', np.float64(0.8098148848579392))
```

We implemented a custom SphericalKMeans estimator compatible with the *scikit-learn* API. The algorithm enforces unit-norm constraints on both the input data and the cluster centroids during optimization. The update steps were optimized following the algorithms described by (Schubert, Lang, and Feher 2021).

Our implementation include:

- Unit Normalization
- Cosine Similarity maximization:
  - The core objective is to maximize the summation of cosine similarities between samples and their assigned centroids.
- The algorithm returns a tree structure. The root node contains all tag embeddings; the next level (depth 1) contains the target centroids (100 in our case); subsequent levels contain increasingly specific tag embeddings.
  - For all target tags, we calculated the mean and identified the 'most representative tag' (the tag closest to the cluster mean).

From this point forward, we refer to the clusters at depth 1 of the recursive tree as the clusters for simplicity sake.

Reduction Analysis

The resulting clusters demonstrated strong semantic coherence. For example, for the outlier centroid with most directly connected tags, the algorithm effectively grouped database-related technologies:

- Cluster Centroid: `database`
- Members: `database`, `mongodb`, `sql`, `elasticsearch`, `firebase`, `postgresql`, `sqlite`, `mysql`, `jdbc`, `supabase`, `sql-update`, `redis`, `hibernate`, `sqlalchemy`, `snowflake-cloud-data-platform`

Similarly, we observed distinct clusters for front-end frameworks (grouping angular, vue.js, react), mobile development (android, flutter, ios), and scientific computing (pandas, numpy, python).

Another impressive result of this reduction is its ability to classify question embeddings solely based on cosine similarity. As an example, we evaluated how well the model classifies the `.net` tag. We computed the cosine similarity between a given question and each cluster, selecting the two closest clusters. We then computed a confusion matrix based on the presence of `.net` in the ground-truth question tags, which yielded the results in fig. 10.

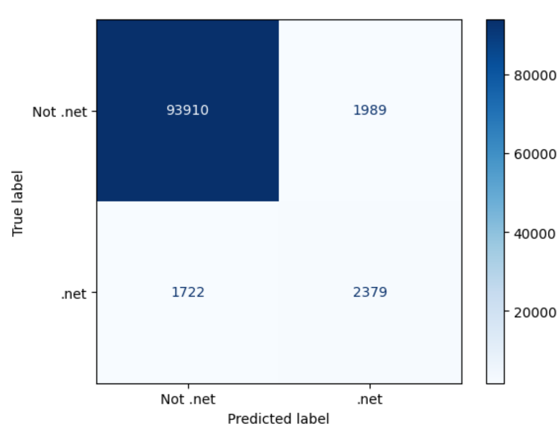

Figure 10: Confusion Matrix of top 2 results for `.net`

Based on this functionality, we've made a basic model `DualEncoderMatcher`, which classifies tags based on cosine similarity mentioned above.

As an experiment, we took random question and checked 5 most similar clusters to it:

- Question:

```
'I am new to pandas library in python. When I loaded a file and was printing the output of df.info
↪  into console, the data ...'
```

- Result:

```
Top 5 predictions: [('python', np.float64(0.38295367797329094)), ('word-table',
↪  np.float64(0.3696944707340029)), ('ironpdf', np.float64(0.3647400168217892)), ('django',
↪  np.float64(0.3297345619706671)), ('apache-spark', np.float64(0.3283243442282066))]
```

These experiments demonstrated the semantic power held by these clusters and prompted us to move towards prediction.

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 body embeddings.

XGBoost Baseline

As a baseline, an XGBoost (Chen and Guestrin 2016) 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

To improve baseline performance, several neural network architectures were explored to better capture the semantic relationships in the embeddings and to natively handle the multi-label nature of the task.

Simple MLP Baseline

We implemented a baseline using a simple Multi-Layer Perceptron (MLP) trained on question body embeddings. Given the multi-label nature of the problem, where a single question may contain both popular tags (e.g., `python`) and rare tags (e.g., `darts`), we had to employ a loss function which takes it into consideration. Initially, we employed `BCEWithLogitsLoss`, which applies a binary cross-entropy loss independently to each class.

While `BCEWithLogitsLoss` is standard for multi-label tasks, it treats all negative labels equally. In a sparse setting where most tags are negative for any given sample, the easy negatives can overwhelm the training signal. Despite these limitations, the MLP baseline achieved a weighted F1 score of approximately 0.6790. This reasonable performance confirmed the viability of the embedding approach but highlighted the need for richer architectures to capture the nuanced relationship between a question's title and its body.

Dual-Stream Fusion Network (DSF)

To better leverage the distinct semantic information contained in titles and bodies, we developed a custom Dual-Stream Fusion (DSF) network, adapting the architecture proposed by Yang et al. for multi-modal fake news detection (Yang et al. 2024).

Architecture The model processes the Title and Body embeddings through two separate streams. These streams are then integrated using a Multi-Head Self-Attention (MHSA) mechanism. The core idea is that the attention mechanism can

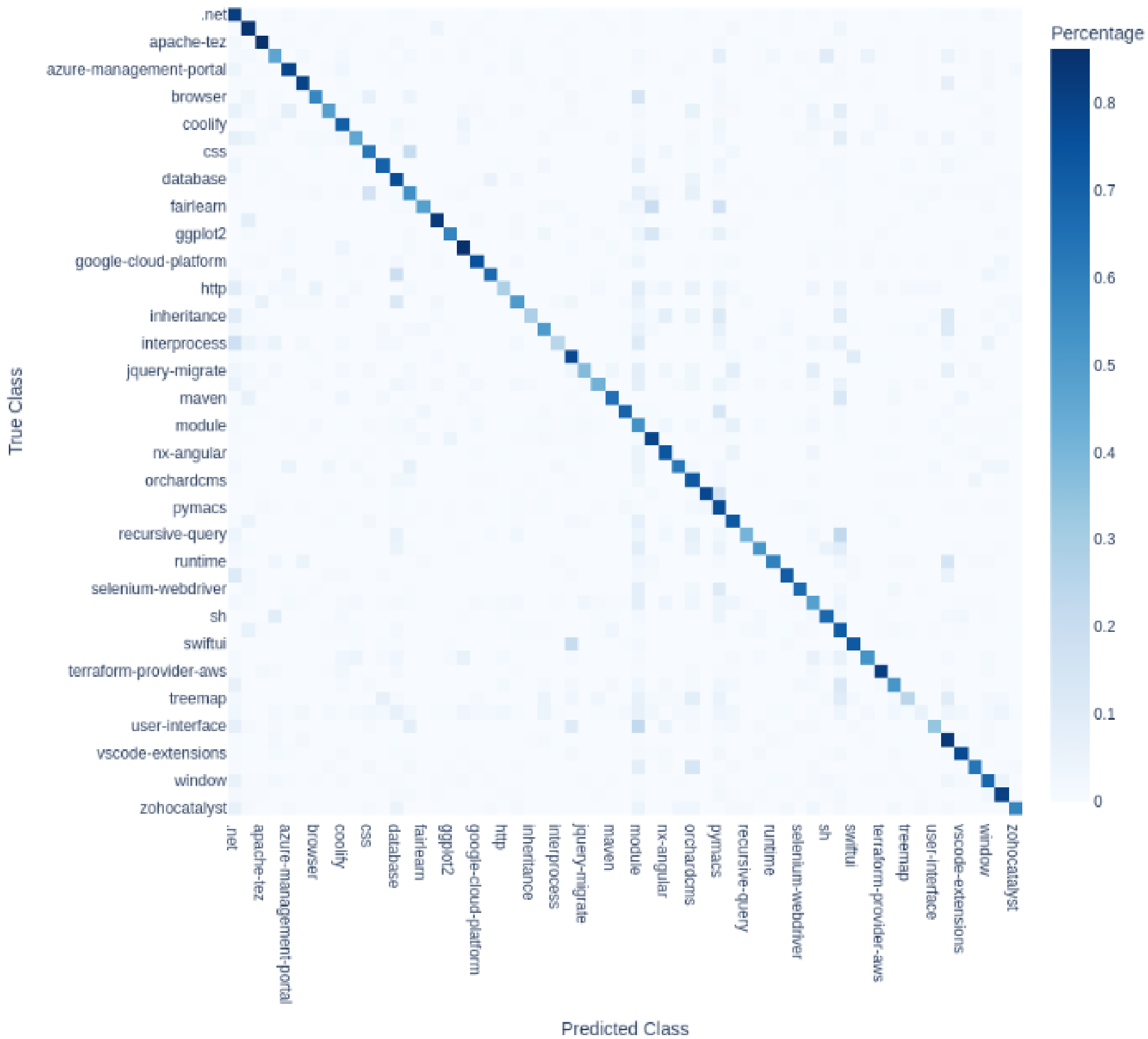

Figure 11: Confusion Matrix of XGBoost Classifier

Figure 12: Validation set performance for MLP baseline model

dynamically weigh the importance of the title versus the body depending on the context. For instance, a short, distinct title might carry more weight than a long, vague body text.

Regularization and Overfitting Initial training runs revealed significant overfitting. The model's validation performance rapidly deteriorated below the MLP baseline after only a few epochs. This demonstrated the "double-edged sword" of attention mechanisms: while they offer high expressivity, they allow the model to easily memorize the relatively small dataset of ~100k samples. Standard regularization techniques, such as increased dropout, weight decay, and aggressive learning rate schedulers yielded limited success.

Notable improvement came from implementing a different loss function, namely `Assymetric Loss` (ASL) (Ben-Baruch et al. 2021). In multi-label classification with many classes, the "negative" samples (tags not present) vastly outnumber the "positive" ones. Standard Cross Entropy allows these easy negatives to dominate the gradient, washing out the signal from the rare positives. ASL addresses this by dynamically down-weighting easy negatives. It introduces two focusing parameters, γ_+ and γ_-. By setting $\gamma_- > \gamma_+$, ASL aggressively suppresses the loss contribution from negative samples the model is already confident about. This forces the optimization process to focus on "hard" negatives (confusing tags) and positive samples, effectively handling the extreme class imbalance without manual re-weighting.

We trained the optimized DSF model with `AdamW` optimizer, `ReduceLROnPlateau` scheduler, and ASL loss with $\gamma_- = 4, \gamma_+ = 1$ for 50 epochs, which resulted in F1 score (micro) on validation set of 0.713, and weighted of 0.711.

Figure 13: Snippet of Per-Tag performance of DSF-MHSA model on the validation set. Each collumn reflects precision, recall f1-score respectively.

As seen in fig. 13, the use of a high γ_- in ASL successfully boosted recall for many classes. However, this comes with a trade-off: simply increasing γ_- indefinitely leads to an overemphasis on recall at the expense of precision. This behavior indicates that while advanced loss functions can mitigate data imbalance, further performance gains likely require increasing the dataset size rather than just architectural tuning.

DSF with Cross-Attention Fusion

To further combat overfitting and improve feature interaction, we evolved the architecture by replacing the Multi-Head Self-Attention (MHSA) with Cross-Attention. In the standard MHSA approach, the concatenated title and body features attend to themselves. However, this treats titles and question bodies equally informative sources.

In our revised Cross-Attention design, we leverage the distinct roles of the inputs: the Title embedding acts as the Query, while the verbose Body embedding serves as the Key and Value. This architectural choice allows the concise, high-density information in the title to "search" the extensive body text for relevant supporting features, effectively filtering out noise from the longer description . Basically, our Cross-Attention layer poses the following question:

> *Given this Title, which parts of the Body Embedding are relevant?*

The projection layers for both streams were standardized using LayerNorm, GELU activation, and Dropout to ensure stable gradient flow before fusion.

Data Mixup To further regularize the model, we implemented Manifold Mixup (H. Zhang et al. 2018). Unlike standard data augmentation which operates on raw inputs, Manifold Mixup constructs virtual training examples by computing convex combinations of pairs of embedding vectors and their corresponding labels. By applying it on the embeddings, we encourage the model to behave linearly in-between training examples, smoothing the decision boundaries and reducing the memorization of outlier data points.

Fine Tuning While the cross-entropy loss optimizes the probability distribution, our evaluation metric (F1 Score) relies on binary decisions. The default decision threshold of 0.5 is rarely optimal for multi-label classification, especially with imbalanced classes. We performed a post-training optimization step, searching for the probability threshold that maximizes the F1 score on the validation set. This calibration ensures that the model's confidence aligns with the optimal precision-recall trade-off.

Training Results

This model, trained over 100 epochs (~60 minutes) using a `OneCycleLR` scheduler, achieved our best performance to date:

- F1 Micro: 0.7253
- F1 Weighted: 0.7196

The DSF with Cross-Attention Fusion mainly has trouble with tags like "import," "installation," and "validation," which probably suggests that it is difficult to distinguish between common technical noise and particular topical intent. The Asymmetric Loss (ASL) may have over-suppressed terms like "import" as "easy negatives" because they appear as boilerplate in nearly every code snippet. Furthermore, the model's inability to handle specialized tags like "asp.net-web-api" indicates that the Cross-Attention mechanism may occasionally lack a detailed enough Title "Query" to extract particular nuances from the verbose Body text.

Sequence-Aware DSF with Cross-Attention

All approaches discussed so far compressed the question body into a single embedding vector before passing it into the network. This operation inevitably results in the loss of details.

To address this, we developed the Sequence-Aware DSF. Instead of single vector, the body stream now processes a sequence of 32 embedding vectors (as mentioned in Sequential Token Embedding). This increase in data fidelity required a redesign of the network architecture.

Architecture Changes:

- Conv1d Projection:
  - We replaced the simple Linear projection with a 1D Convolutional layer. This allows the model to capture local n-gram-like patterns within the sequence of embeddings.
- Self-Attention Pre-Processing:
  - Before fusion, the body sequence passes through a Self-Attention layer. This allows the model to construct a "global body context," relating distant parts of the text (e.g., an error message at the bottom to a code snippet at the top).
- Cross-Attention Fusion:
  - The Title (Query) attends to this refined Body Sequence (Key/Value), selecting the specific tokens most relevant to the question summary.

FocalLoss and Label Smoothing

With the increased complexity of the Sequence-Aware model, we observed that AsymmetricLoss (ASL) and BCEWithLogitsLoss resulted in unstable training dynamics. The model tended to oscillate or converge to suboptimal minima. To stabilize optimization, we adopted Focal Loss (Lin et al. 2018).

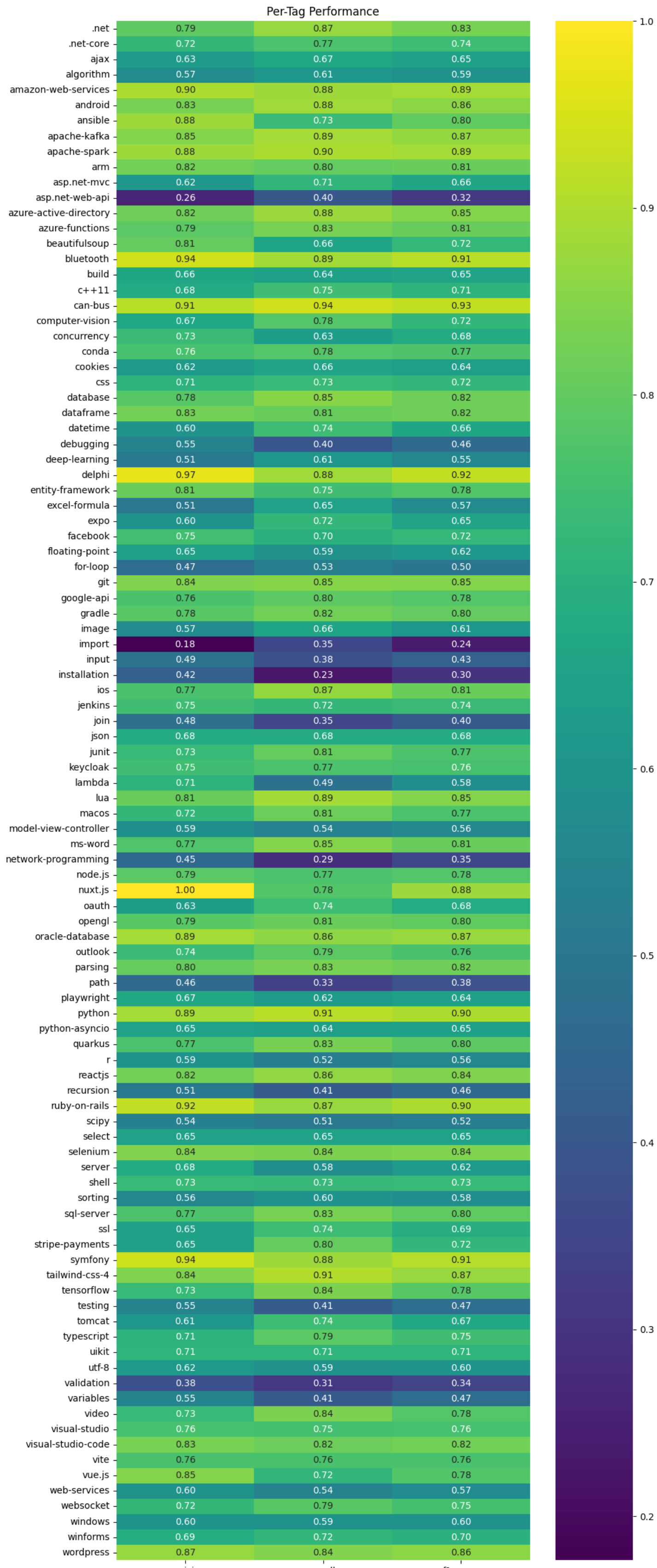

Figure 14: Per tag performance of DSF with Cross-Attention Fusion

Focal Loss reshapes the standard cross-entropy loss to down-weight easy examples and focus training on hard negatives. By reducing the loss contribution of easy examples, the model is forced to learn the difficult, ambiguous cases that are common in the dataset.

We further refined this by implementing *Focal Loss with Label Smoothing*. Standard one-hot targets (0 or 1) encourage the model to be overconfident, pushing logits towards infinity. Label smoothing relaxes these targets, and prevents overfitting by penalizing overconfidence, resulting in better generalization on the validation set.

Training Results

Unfortunately, our models proved "too strong" for the available data. The networks easily memorized the training set, even with the aforementioned regularization techniques and custom loss functions. Ultimately, FocalLoss performed worse than BCEWithLogitsLoss (with class weights). FocalLoss with Label Smoothing yielded slightly better results but still underperformed our simple MLP baseline, achieving an F1 Micro of 0.6441 and F1 Macro of 0.615.

An important observation during the training phase was the sensitivity of the decision threshold. As detailed in the Fine Tuning section, the basic FocalLoss required a very high optimal threshold (≈ 0.9), indicating that the model was overly confident in its predictions (logits pushed to extremes). in contrast, FocalLoss with Label Smoothing resulted in a more balanced optimal threshold of 0.4.

Throughout training, we consistently hit a "Generalization Ceiling." While the training loss approached near-zero (0.003), the validation F1 score remained stubbornly stuck at ≈ 0.64. Neither Manifold Mixup nor extensive fine-tuning yielded significant improvements.

This behavior highlights a fundamental limitation often observed in transformer-based architectures: their lack of inductive bias. Unlike Convolutional Neural Networks, which have built-in assumptions about locality and translation invariance, or MLPs, which are structurally simpler, attention mechanisms are extremely flexible. This flexibility allows them to learn complex global relationships but also makes them highly "data-hungry" (Dosovitskiy et al. 2020; d'Ascoli et al. 2021). Without massive datasets to constrain the search space, transformers tend to overfit the noise in small-to-medium datasets (like our 100k samples) rather than learning robust generalized features.

Seq2Seq Model

Instead of defining our task as a classification problem, we can rephrase it as a Sequence to Sequence problem, where we map one sequence (question text) into another (tag tokens). This approach holds significant potential, as we can leverage the unprecedented capabilities of LLMs by taking a pre-trained model (in our case `t5-small`) and fine-tuning it for our task.

The biggest advantage is also it's biggest disadvantage, all modern models require prohibitively large amount of compute, which forced us to pick a relatively small (60M) model. We picked `t5-small` because it was trained primarily on summarization and translation tasks, which closely aligns with our goal of "summarizing" a question into tags.

After 5 hours of training, we've achieved F1 micro score of 0.6676 and F1 macro of 0.625. While this may not sound as impressive as previous models, we must consider the size of this model and the transformers' innate need for huge amounts of data.

Another advantage of this model is the ease of use in terms of a user-readable format. Using `transformers`, we can easily create a quick prediction function for any given question:

```
t = "How do I reverse a list?"
b = "I have a list [1, 2, 3] and I want [3, 2, 1]. slicing doesn't work for me. I'm thinking of
↪ using a pandas library "
print(predict_custom_question(t, b))
dataframe, python, sorting
```

A powerful feature of this approach is that the model itself infers how many tags are needed for each question, rather than relying on a fixed top-k or threshold.

Summary of Results

As shown in tbl. 3, the DSF with Cross-Attention Fusion emerged as the superior model, achieving a Weighted F1 score of 0.7196.

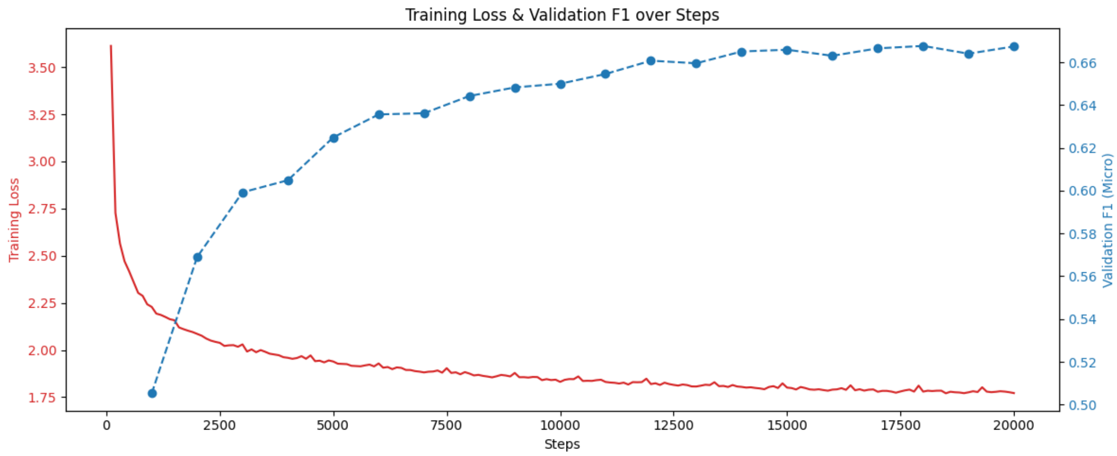

Figure 15: Training Loss and Validation F1 over Steps

Table 3: Summary of Tag Prediction Results.

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**

Performance Analysis

While the XGBoost seems to outperform Baseline MLP, we have to keep in mind, that XGBoost tackled much simpler task (mere single-class classification) and required more time to compute than all other models combined.

The significant leap in performance came with the Dual-Stream Fusion (DSF) architectures. The introduction of Cross-Attention proved critical; by treating the Title as a "Query" and the Body as a "Key/Value," the model successfully filtered verbose noise from the body text. This approach, combined with Asymmetric Loss (ASL) to handle class imbalance and Manifold Mixup for regularization, allowed the model to generalize better than the Multi-Head Self-Attention (MHSA) variant.

The more complex Sequence-Aware model and Seq2Seq in our opinion underperformed, due to a fundamental limitation often observed in transformer-based architectures: their lack of inductive bias. Unlike Convolutional Neural Networks, which have built-in assumptions about locality and translation invariance, or MLPs, which are structurally simpler, attention mechanisms are extremely flexible. This flexibility allows them to learn complex global relationships but also makes them highly "data-hungry" (Dosovitskiy et al. 2020; d'Ascoli et al. 2021). Without massive datasets to constrain the search space, transformers tend to overfit the noise in small-to-medium datasets (like our 100k samples) rather than learning robust generalized features.

Our dataset, while seemingly large, was insufficient for a complex attention-based model to learn the intricate semantic mappings required for multi-label tagging without falling into the trap of memorization.

Score Prediction

Our secondary regression task focused on predicting the question score, which reflects community engagement and perceived quality. 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).

Traditional ML Baseline

To establish a performance floor, we conducted an initial exploration using traditional NLP techniques, combining TF-IDF feature extraction with Truncated SVD (Latent Semantic Analysis) for dimensionality reduction. We benchmarked several configurations, including:

- Linear Models: TF-IDF (with variations in n-grams and stemming) paired with Ridge Regression.
- Ensemble Methods: Gradient Boosting (GBR) utilizing TF-IDF, SVD, and engineered features.
- Baseline Comparisons: Standard Bag-of-Words (BoW) and a simple mean-prediction baseline.

The results were underwhelming:

- 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.

We also briefly experimented with $log1p(score)$ targets (LinearSVR/SGD variants) to soften the extreme heavy tail, but the models merely learned to linearize the head of the distribution and lost the already fragile fidelity on rare high-score posts. This near-zero R^2, regardless of the target scaling, underscores the inherent difficulty of the task: traditional frequency-based features are insufficient for capturing the complex, non-linear relationships that drive community engagement (scores) on Stack Overflow. This served as a strong justification for moving toward the deep learning approaches detailed below.

Deep Learning Regressors

To evaluate the predictive power of neural architectures on numerical outcomes, we adapted the single-stream MLPs and the dual-stream DSF variants for regression by utilizing a Mean Squared Error (MSE) loss function. In addition to the semantic embeddings, we integrated a normalized, clipped tag count feature to provide the model with a proxy for question complexity.

As shown in tbl. 4, all deep learning configurations achieved a substantial performance leap over the traditional baselines ($R^2 \approx 0.000$).

Table 4: Score Regression Performance on Test Set (using Embeddings).

Model	Input	Test RMSE	Test R²
Mean Baseline	Constant	149.97	0.000
Title MLP	Title Embedding Only	138.73	0.144
Body MLP	Body Embedding Only	137.57	0.158
DSF-CrossAttention Regressor	Title + Body Embeddings	135.27	0.186
DSF-MHSA Regressor	Title + Body Embeddings	134.24	**0.199**

The DSF-MHSA Regressor emerged as the top performer, explaining approximately 19.9% of the score variance. Interestingly, in the regression context, the global focus of Multi-Head Self-Attention (MHSA) slightly outperformed the more targeted Cross-Attention mechanism.

The factor that explains the MHSA outperforming the rest of the models is mostly attributed to how MHSA pools both streams at once. Long-distance hints about wording quality, structure, or clarity can be mixed without forcing one stream to "query" the other, so the clipped tag-count signal is tied back to the overall question narrative more reliably. By contrast the Cross-Attention variant was occasionally locked onto short title fragments while the body was ignored. In repeated runs the MHSA curves were also observed to be smoother on the validation set, which matches the small but consistent RMSE gap in tbl. 4.

Despite the improvement, diagnostic analysis revealed a common challenge in social data regression: while the model accurately predicts the vast majority of low-scoring posts, it struggles to capture the "viral" outliers or high-score peaks (fig. 18). This suggests that high scores may be driven by external temporal factors or community dynamics not fully captured within the text embeddings alone.

While conducting the experiments we trained both DSF regressors for the full 50 epochs without early stopping. The learning curves in fig. 16 show how quickly the validation loss diverges from the training loss, reinforcing the decision to keep a patience-based early stopping heuristic in the final runs (without it the DSF-MHSA RMSE degraded to 147.7 despite a seemingly stable validation loss around epoch 10).

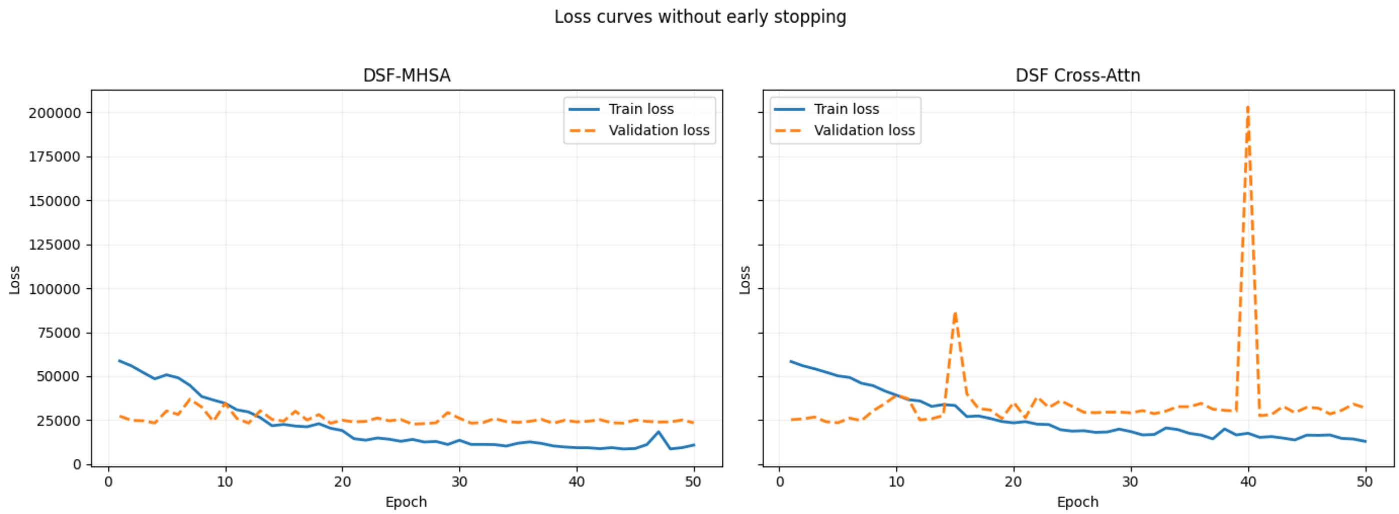

Figure 16: Loss curves for DSF-MHSA and DSF Cross-Attention without early stopping.

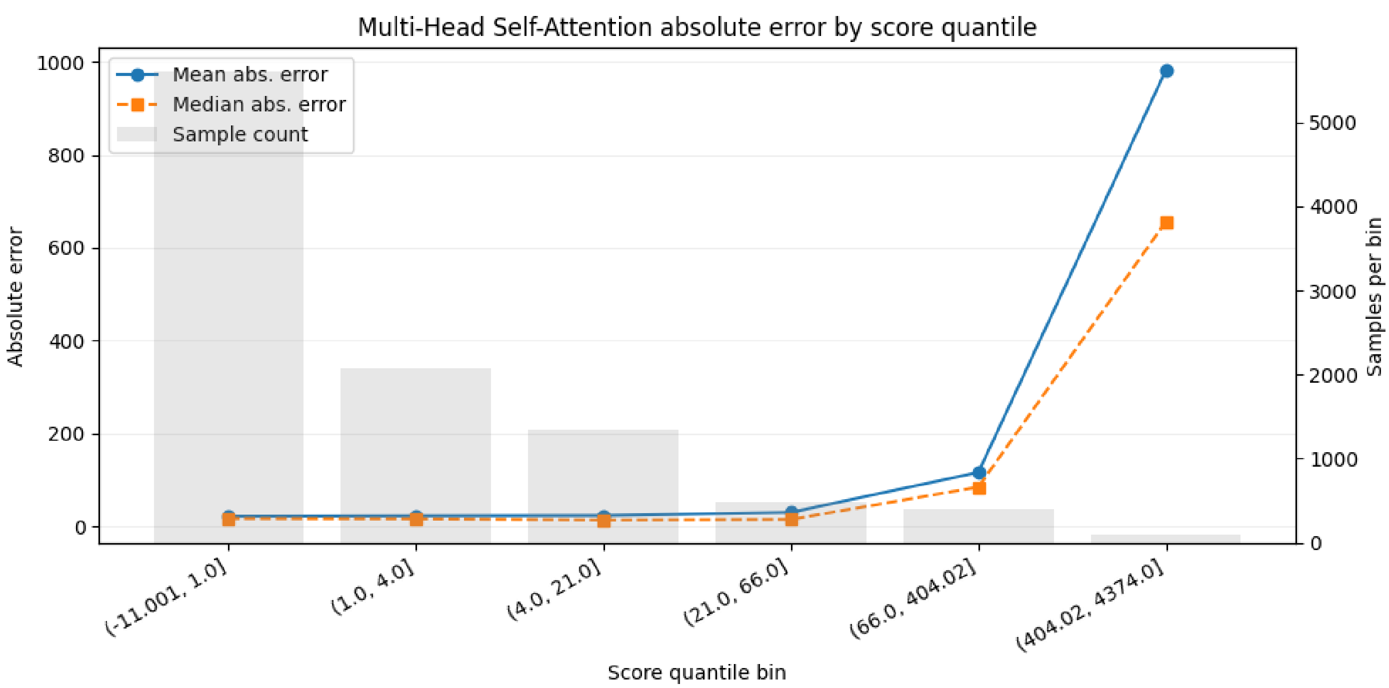

Figure 17: Cross-Attention absolute error by empirical score quantiles; bars show sample counts per bin.

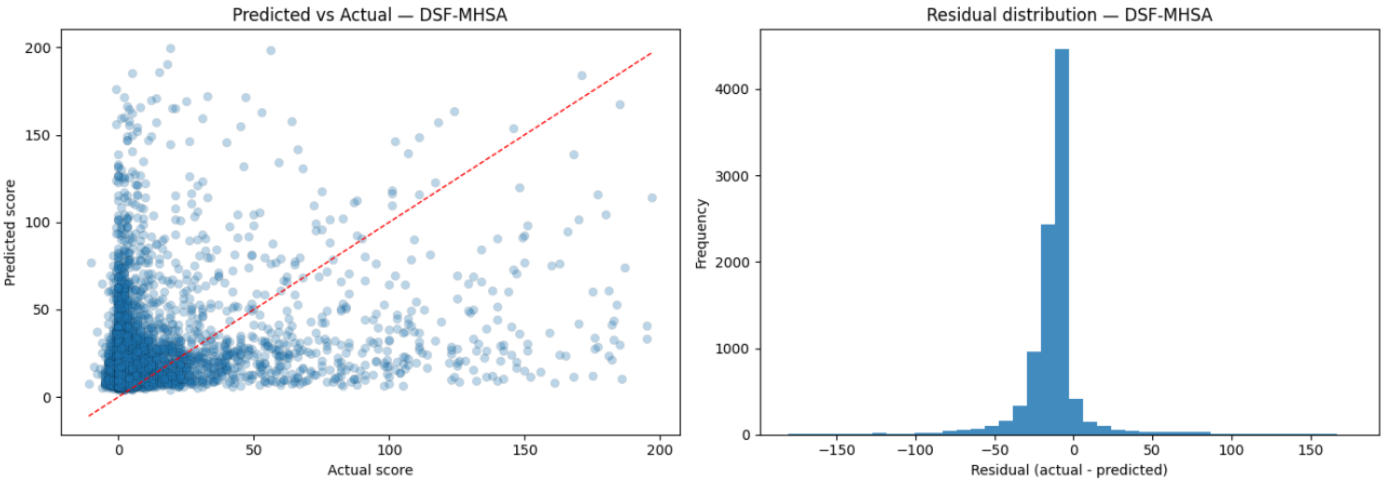

Figure 18: Predicted vs Actual Scores (DSF-MHSA, Filtered Range)

Conclusions and Future Work

In this project, we explored the viability of utilizing embeddings for the multi-label classification and score prediction of StackExchange questions. Our investigation progressed from unsupervised dimensionality reduction to complex, attention-based neural network architectures.

We successfully demonstrated that the extreme cardinality of the tag space (over 22,000 unique labels) could be effectively managed through **Recursive Spherical K-Means Clustering**. By organizing tags based on semantic cosine similarity rather than Euclidean distance, we reduced the target space to 100 dense, semantically coherent centroids without significant loss of information. This hierarchical structure proved superior to flat clustering methods and provided a robust foundation for classification.

Our clustering algorithm also proved, that 'off-the-shelf' state-of-the-art algorithms (such as UMAP or HDBSCAN) are not universally optimal. By looking beyond standard scikit-learn implementations and adapting methodologies from engineering blogs and academic literature, we developed a custom solution, which not only solved the memory constraints that plagued standard algorithms but also provided superior computational efficiency for our specific high-dimensional embedding space.

Key Successes and Insights

- **Robust Tag Reduction:** The greatest technical achievement in the data preparation phase was the establishment of the **Recursive Spherical K-Means** algorithm, 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.
- **Inductive Bias vs. Data Volume:** The ultimate limitation observed across our experiments in tag prediction, was the trade-off between model flexibility and dataset size. As noted in Performance Analysis, Transformer-based attention mechanisms lack the inductive bias of simpler architectures (like CNNs or MLPs). They are extremely flexible but highly "data-hungry".
- Our dataset of $\approx$ 100,000 questions was insufficient to constrain the vast search space of the fully sequence-aware models, leading them to memorize noise rather than learn robust generalized features. Consequently, the DSF with Cross-Attention represented the optimal "sweet spot": it was complex enough to model semantic interactions via attention, but structured enough (compressing the body into a single embedding) to avoid the overfitting pitfalls of full sequence modeling.
- **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

1. 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.
2. Dataset Expansion and Augmentation: For classification, acquiring dataset of size around 10000000 could drastically enhance performance beyond the current pooled embedding methods.
3. Hierarchical Classification: Currently, our model predicts the target clusters directly. A more refined approach would leverage the tree structure generated by our Recursive K-Means algorithm.
    - We propose a Hierarchical Classification pipeline, where a model first predicts the broad category (e.g., *Web Development*) and subsequent heads predict specific frameworks (e.g., *React*, *Angular*).
4. Graph Neural Networks: Our current approach treats labels as independent targets (except for the implicit grouping in centroids). Implementing Graph Neural Networks or Graph Convolutional Networks to model the explicit dependencies between tags could significantly improve prediction accuracy, particularly for correlated technologies.

References

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. "Optuna: A Next-Generation Hyperparameter Optimization Framework." In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ben-Baruch, Emanuel, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. 2021. "Asymmetric Loss for Multi-Label Classification." https://arxiv.org/abs/2009.14119.

Caliński, T., and J Harabasz. 1974. "A Dendrite Method for Cluster Analysis." *Communications in Statistics* 3 (1): 1–27. https://doi.org/10.1080/03610927408827101.

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–94. KDD '16. ACM. https://doi.org/10.1145/2939672.2939785.

d'Ascoli, St'ephane, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. 2021. "ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases." In *International Conference on Machine Learning*, 2286–96. PMLR. https://arxiv.org/abs/2103.10697.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al. 2020. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale." *arXiv Preprint arXiv:2010.11929*. https://arxiv.org/abs/2010.11929.

Hornik, Kurt, Ingo Feinerer, Martin Kober, and Christian Buchta. 2012. "Spherical k-Means Clustering." *Journal of Statistical Software* 50 (10): 1–22. https://doi.org/10.18637/jss.v050.i10.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. "Focal Loss for Dense Object Detection." https://arxiv.org/abs/1708.02002.

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.

Schubert, Erich, Andreas Lang, and Gloria Feher. 2021. "Accelerating Spherical k-Means." In *Similarity Search and Applications*, 217–31. Springer International Publishing. https://doi.org/10.1007/978-3-030-89657-7_17.

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

Yang, Yimei, Jinping Liu, Yujun Yang, and Lihui Cen. 2024. "Dual-Stream Fusion Network with Multi-Head Self-Attention for Multi-Modal Fake News Detection." *Applied Soft Computing* 167: 112358. https://doi.org/https://doi.org/10.1016/j.asoc.2024.112358.

Zhang, Hongyi, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. "Mixup: Beyond Empirical Risk Minimization." https://arxiv.org/abs/1710.09412.

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.