Data preparation

As a first step, I extracted the diff for all commit #hashes in the previous dataset. This process was time-consuming due to the nearly 120,000 rows, resulting in a final dataset `final_dataset_with_diffs.csv` of 26.7GB.

The extraction took several iterations because the system struggled and failed many times when attempting a direct approach.

1. **Batch-wise approach:** This failed because some diffs were too large. ❌
2. **Store one by one to MongoDB (using multi-threading for speed):** This failed due to the maximum document size limit of 16MB, which some diffs exceeded.❌
3. **Directly store to CSV one by one:** This method was slow but eventually worked, taking several days to complete. ✅

Tokenizing

The laptop can't handle such large data set even can't visualize it
I just upload the data set to the drive and open in google colab v2-8 TPU environment

Problem 1
While tokenizing the code this make trouble me with size and some null values
So first of all I need to clean the data and separate large size data records so I can process them separately (to solve the memory bottleneck)

Problem 2
Each diff are different each other so the token size will vary (to train a xgboost like model the final dataset should have equal columns for each row) each token will returns a vector (we call this embeddings)
That mean each diff return more than 1 embeddings so to get the overall context we should use a pooling mechanism **Attention-Based Pooling (Best for Semantic Depth)**, **CLS Token Embedding**,**Mean Pooling**

To reduce the load and extractor computation power I use the **CLS Token Embedding**

Before tokenizing, a special classification token, `[CLS]`, is added to the beginning of every diff. The CodeBERT model is specifically designed to process the entire sequence and compress its overall meaning into the final embedding of this single `[CLS]` token.

Problem 3

The CodeBERT model's maximum token length is 512 When a diff is longer than the model's 512-token limit and all parts are important, simple **truncation** is not a good option because you lose valuable information.

The standard and most effective way to handle this is a technique called **Sliding Window with Pooling**.

How Sliding Window with Pooling Works

Instead of just cutting off the diff, you process it in manageable pieces and then combine the results.

1. **Divide into Overlapping Chunks**: The long diff is broken down into smaller chunks, each 512 tokens long. Crucially, these chunks **overlap** (e.g., the last 100 tokens of the first chunk are the first 100 tokens of the second chunk). This overlap ensures that the model sees the context at the boundaries of each chunk.
2. **Embed Each Chunk Independently**: The CodeBERT model is run on every single chunk. This gives you a separate embedding for each piece of the long diff. For example, a 3000-token diff might be broken into 6 or 7 chunks, resulting in 6 or 7 different embedding vectors.
3. **Combine the Embeddings (Pooling)**: The final and most important step is to combine these multiple chunk embeddings into a single vector that represents the entire diff. The most common method is **Pooling**, where you simply average the vectors of all the chunks. This creates a final, single embedding that incorporates information from every part of the original long diff.

This approach ensures that every line of the diff contributes to the final embedding, solving the problem of information loss from truncation.

B82855a671835fe6b0f4fc5ffd3e65a30312e620 is problematic data row
In filter data data row 2006 is problematic filtered_output.csv