Project Walkthrough: Building a System to Detect Coordinated Cyberbullying

Introduction: Beyond Single Messages

Traditional content moderation often operates on a message-by-message basis, asking "Is this specific post harmful?" While useful, this approach can miss a more insidious form of online harm: **coordinated harassment**. This is "crowded" cyberbullying, where a target is overwhelmed by a pile-on of many messages, often from multiple users in a short period. A single message might be ambiguous, but a hundred similar messages are a clear attack.

The goal of this project is to build a fast inference system that can look beyond individual posts and analyze **groups of messages** as a whole. By understanding the dynamics within a set of recent comments, the system can spot dense, concentrated attacks as they happen and provide real-time alerts.

This walkthrough will guide you through the four-stage pipeline developed to achieve this goal:

1. **Stage 1: Teaching a Model to Spot Bullying:** Training a foundational classifier to understand the content of individual messages.
2. **Stage 2: Capturing Semantic Meaning:** Creating numerical "fingerprints" (embeddings) for each message to understand what they *mean*.
3. **Stage 3: Analyzing Group Dynamics:** Building advanced models that can score an entire *set* of messages to identify group-level risk.
4. **Stage 4: Building the Real-Time Alert System:** Simulating a live environment to detect and explain coordinated attacks as they unfold.

Let's begin with the first essential building block: a model that can recognize a single piece of bullying text.

---

1. Stage 1: The Foundation - A Message-Level Bullying Classifier

Before we can analyze groups, we first need a reliable way to score individual messages. The goal of this stage is to build a fundamental component—a model that can assign a "bullying confidence" score to any piece of text.

Choosing the Right Tool for the Job

The first step was selecting a suitable dataset and model architecture.

The chosen dataset was `cike-dev/cyberbullying_dataset`, a collection of texts labeled as either "bully" or "normal." This choice was strategic for several reasons:

Reason for Choosing This Dataset	What This Means for the Project
Clean, Binary Labels	The "bully" vs. "normal" categories provide a clear and simple starting point for training a baseline classifier.
Sufficient Scale	With over 30,000 text samples, the dataset is large enough to effectively fine-tune a powerful transformer model.
Balanced Classes	The dataset is split evenly (50/50) between bully and normal examples, which simplifies the initial training and evaluation process.

For the model, `DistilBERT` was selected. While larger models exist, DistilBERT offers an excellent balance of performance and efficiency, which is critical for the project's goal of building a *fast* inference system.

Model	Size	Speed	Relative Performance
BERT-base	110M params	Baseline	100% (Baseline)
DistilBERT	**66M params**	**~60% Faster**	**Retains ~97% of accuracy**

This trade-off makes DistilBERT the ideal choice for a first-stage component in a real-time detection pipeline.

Training and Results

According to the final validation classification report, the fine-tuned DistilBERT model achieved strong performance with an accuracy of **0.8061** and an F1-score of **0.8156** for the "bully" class.

Crucially, the output of this model isn't just a simple label. For every message, it produces a `bullying_confidence` score—the probability that the message constitutes bullying. This continuous score is a much richer signal than a binary flag and becomes the foundation for all subsequent stages.

Here is the model in action on two examples:

Input: `you are such an idiot` **Prediction:** `bully` (Confidence: ~0.945)

Input: `nice photo!` **Prediction:** `normal` (Confidence: ~0.046)

With individual message scores established, the next logical problem is to determine if messages are semantically related, which requires converting them into a comparable format: embeddings.

---

2. Stage 2: Capturing Meaning - Creating Semantic Fingerprints (Embeddings)

To detect coordinated harassment, where attackers often repeat similar insults, the system needs to understand the semantic relationship between messages. This is achieved by converting text into numerical vectors called **embeddings**. Think of an embedding as a "semantic fingerprint"—a series of numbers that captures the core meaning of a piece of text.

What Are Embeddings and Why Do We Need Them?

By representing messages as vectors in a high-dimensional space, we can mathematically compare them. Messages with similar meanings will have vectors that are "close" to each other. This is the key to identifying groups of semantically related comments.

For this task, the `all-MiniLM-L6-v2` model was chosen. This model is ideal for two main reasons:

- It is highly optimized for creating **high-quality sentence embeddings**, meaning its fingerprints accurately capture the nuances of text.
- It is **very small and fast**, which is essential for a system designed to process thousands of messages in real-time.

The Output: A Map of Meaning

The output of this stage is a large numerical array (`minilm_embeddings.npy`), where each row is the 384-dimensional vector representing a single message from the dataset.

An important finding from this stage came from a simple experiment: attempting to cluster these embeddings into two groups ("bully" and "normal") yielded a very low silhouette score of approximately 0.032. For a learner, this means that the concepts of bullying vs. normal text are not simple, separate clouds in semantic space. They are complex and overlapping, reinforcing the need for a more advanced model to find meaningful patterns.

With a numerical fingerprint for every message, we can now move from analyzing individual texts to analyzing entire *groups* of them.

---

3. Stage 3: The Core Innovation - Analyzing Group Dynamics

This stage represents the most significant leap in the project. Here, the focus shifts from asking, "Is this message bad?" to a much more powerful question: "**How risky is this entire group of messages?**" This change in perspective is the key to detecting coordinated raids and pile-on harassment.

Attempt 1: A Sanity Check with Deep Sets

The first approach used a model called **Deep Sets** as a simple baseline. In essence, this model looks at all the message embeddings in a given set and outputs a single score representing the group's *average* bullying intensity. This model was tested on synthetically created clusters of messages to prove that the core idea—scoring a set of texts—was viable.

Attempt 2: A Smarter Model with Set Transformer

The next step was to use a more powerful model called a **Set Transformer**. Unlike Deep Sets, which just averages features, the Set Transformer uses an attention mechanism to understand the relationships *between* messages within a group.

Feature	DeepSets	Set Transformer
Understands Relationships	❌ No	✅ Yes
Logic	Averages element features	Uses attention to weigh interactions *between* elements

The Set Transformer's ability to capture complex internal patterns makes it far better suited for identifying nuanced group dynamics than a simple averaging model.

The Key Insight: The Journey to Measuring "Concentration"

Through experimentation, a critical insight emerged: predicting the *average* risk of a group is not enough. A group of 10 mildly negative messages might have the same average score as a group containing 9 perfectly normal messages and one extremely toxic, targeted attack. To a human moderator, the second case is far more alarming, but a model predicting the mean would miss it.

This led to an iterative search for a second, more powerful signal that could capture this "spikiness." The first attempt was to predict `mean(top_k(confidence))`, focusing on just the most toxic messages. This was an improvement but was still fundamentally a measure of

magnitude. The next idea was a `contrast` score (`mean(top_k) - mean`), which tried to measure how much the peak harm stood out from the baseline.

The real breakthrough came with the introduction of the **Gini coefficient**. Borrowed from economics, the Gini coefficient is a measure of inequality or distribution. A high Gini score means that toxicity is highly **concentrated** in just a few messages, while a low score means it is evenly spread out. This was the perfect mathematical tool to describe the difference between a simmering, negative conversation and a sudden raid.

This discovery led to the development of a **dual-head model**. Instead of predicting one score, this model is trained to predict two separate scores for each group of messages:

- **Mean Head:** Predicts the *overall average risk* of the group. This tells us the general "temperature" of the conversation.
- **Concentration Head:** Predicts the Gini coefficient of the toxicity scores. This tells us how "uneven" or "spiky" the harm is within the group.

By predicting both **mean risk** and **concentration**, the system can now distinguish between a low-level, sustained negative conversation and a sudden, highly-concentrated "raid" by a few bad actors. This is the key to detecting coordinated attacks.

With a powerful set-based model now trained to understand group dynamics, the final step is to deploy it in a simulated live environment.

---

4. Stage 4: The Alert System - Detecting Attacks in Real-Time

This final stage brings all the previous components together into a functioning detection pipeline. Using a second dataset of comments (`civil_comments`), this part of the project simulates a live feed of messages to test the system's ability to generate real-time alerts.

Simulating a Live Feed

The system works by maintaining a **rolling context buffer**. This is simply a sliding window that keeps track of the last `N` messages that have come through the stream. At each step, the dual-head Set Transformer model analyzes the current set of messages in this window to produce a continuous assessment of risk.

From Model Scores to Actionable Alerts

The raw `mean_hat` and `conc_hat` scores from the model are first smoothed into `rolling_mean` and `rolling_conc` signals to reduce noise. Critically, `rolling_conc` is calculated as a **rolling maximum** over the raw `conc_hat` scores, which allows it to capture and hold onto the peak of a recent bursty event.

The system then uses these signals to generate two primary types of alerts:

- **`burst` alert:** This alert fires when the `rolling_conc` signal spikes above a threshold. It is designed to catch **sudden, concentrated raids** where a few highly toxic messages appear in a short time.
- **`sustained` alert:** This alert fires when the `rolling_mean` signal remains high for a period. It is designed to catch **prolonged, simmering negativity** in a conversation.

The following table shows a sample of alerts generated by the system. Note how the `rolling_conc` value remains constant across all three alerts; as a rolling max, it captures the peak concentration from the initial event at `msg_id` 4 and holds it as the context window slides forward.

msg_id	rolling_mean	rolling_conc	alert_reason
4	0.0587	0.3961	burst
5	0.1115	0.3961	burst
6	0.1590	0.3961	burst

Explainability: Understanding *Why* an Alert Fired

One of the most valuable features of this system is its ability to provide context. When an alert fires, it doesn't just raise a flag; it can immediately show the specific messages within the context window that contributed most to the high score.

For instance, when the first `burst` alert fired (at `msg_id` 4), the system identified the following messages as the key contributors. Notice how one highly toxic message is surrounded by normal ones—a classic "spiky" pattern that the concentration score is designed to detect.

msg_id	toxicity	snippet
4	0.89	haha you guys are a bunch of losers.

3	0.00	Is this something I'll be able to install on my site? When will you be releasing it?
2	0.00	This is such an urgent design problem; kudos to you for taking it on. Very impressive!
1	0.00	Thank you!! This would make my life a lot less anxiety-inducing. Keep it up, and don't let anyone get in your way!
0	0.00	This is so cool. It's like, 'would you want your mother to read this??' Really great idea, well done!

This concludes the walkthrough of the complete pipeline, from training a basic classifier to building a sophisticated, explainable real-time alert system.

---

5. Conclusion: Key Learnings and Future Directions

This project demonstrates a novel approach to cyberbullying detection by shifting the focus from individual messages to group dynamics. The journey provides several key insights for anyone interested in building similar systems.

Summary of Findings

- **Message-level models are a good start:** A standard `DistilBERT` classifier provides a solid baseline signal for message toxicity, achieving a `bully` F1-score of over **0.81**.
- **Group dynamics are complex:** Simply clustering messages by their semantic meaning is not enough to cleanly separate "bully" conversations from "normal" ones. The relationship is more nuanced.
- **Set-based models are powerful:** Architectures like the Set Transformer, which analyze entire groups of messages, are highly effective for detecting coordinated behavior.
- **Concentration is a critical signal:** The project's most significant insight is that measuring the *concentration* of toxicity (using a metric like the Gini coefficient), not just the average, is essential for identifying raid-like harassment.

Limitations and Next Steps

The current system has some limitations, such as its reliance on a binary "bully/normal" label and the use of simulated rather than real timestamps for message arrivals.

A powerful next step would be to incorporate **per-thread aggregations**. Instead of analyzing the last `N` messages from all conversations, the system could track messages within specific conversation threads. This would make the context more precise and **reduce false positives caused by mixing unrelated conversations into the same context buffer**, creating an even more robust and intelligent detection system.