ML System Design Harmful Content Detection System

The video is available at https://www.youtube.com/watch?v=Tjs23TCMQIQ

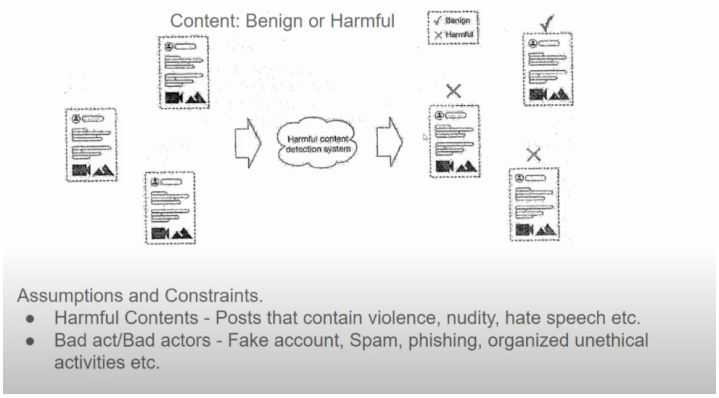
1. **Objective**: to detect whether a given post is harmful or not. The process starts with an incoming post from a user. This post can contain text, images, videos, or a combination of media.

Importance of This Process for Social Media Platforms

Social media websites benefit greatly from this type of content filtering because harmful content (e.g., hate speech, misinformation, explicit content) can lead to negative outcomes, such as user distress, brand damage, and even regulatory fines. Since majority of money comes from ads which are placed by brands, the companies might not want to advertise if their name is associated with harmful content. By automating harmful content detection, platforms can:

1. **Protect Users**: Reduce exposure to harmful or distressing content, helping create a safer online environment.
2. **Ensure Compliance**: Avoid regulatory repercussions by adhering to policies and laws around harmful content.
3. **Maintain Reputation**: Protect the platform's brand by preventing association with harmful or offensive material.
4. **Enhance Engagement**: Increase trust among users, encouraging engagement and content sharing in a positive environment.

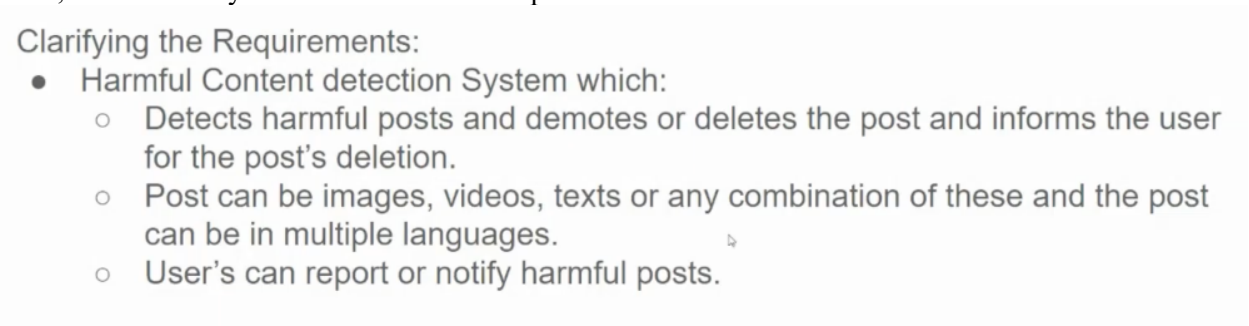
Automating this process also enables social media platforms to handle the massive scale of user-generated content efficiently, which would be unmanageable with human moderation alone.



In this case we will design a system that only detects harmful contents. The objective to detect bad actors is OOS. One way to do this - We can have human annotators that can do this, but this cannot scale. So, we turn to using an ML system.

2. **Clarifying the Requirements**

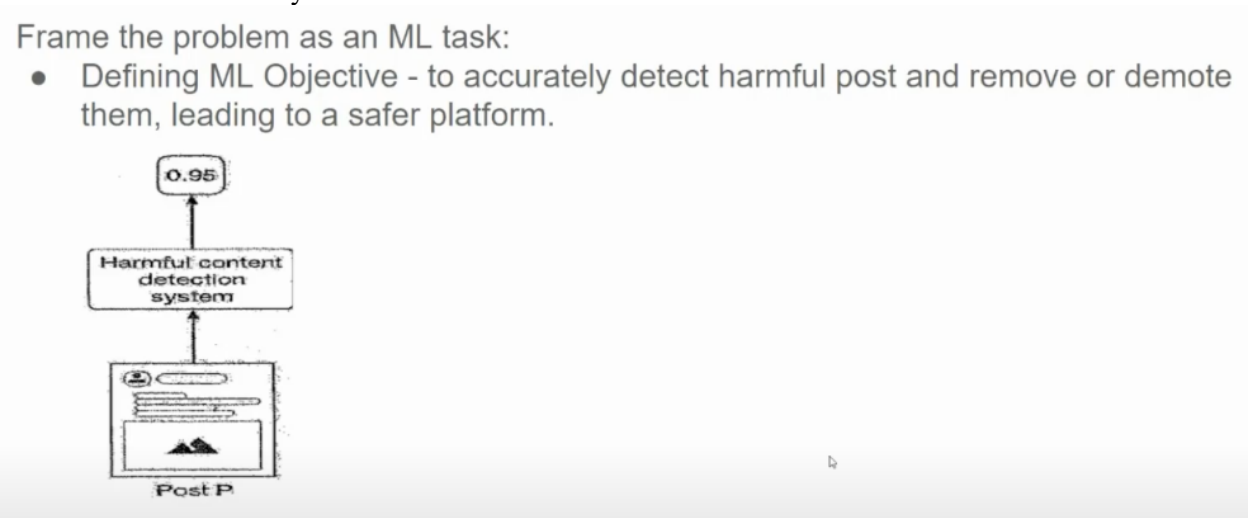
Next, we will clarify what exact is a harmful post



3. **Framing as ML problem**

Next, it is important to frame this as an ML problem. It might fall into one of the following

- Classification
- Regression
- Recommender System



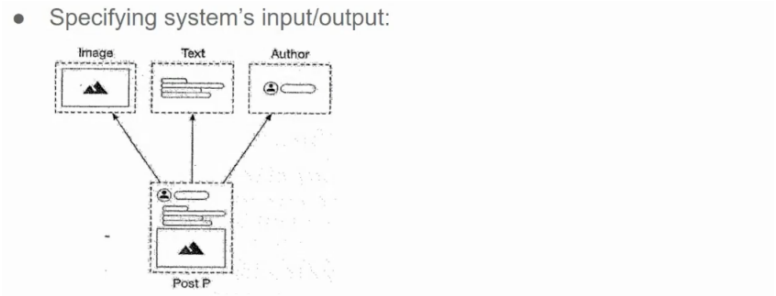
So, we can infer that this is a classification task

4. **Data**

Flow Description:

1. **Input Post**: The process starts with an incoming post from a user. This post can contain text, images, videos, or a combination of media.
2. **Preprocessing**: The post is analyzed and preprocessed to make it ready for further inspection. This step may involve:
    - **Tokenization** of text to break it down into words or phrases.
    - **Filtering** and **normalizing** content (e.g., removing emojis, converting text to lowercase).
3. **Feature Extraction**:
    - For text: Keywords, sentiment, and tone are extracted to understand the context and detect indicators of harmful content.
    - For images: Objects, context, or sensitive visual content are detected, sometimes using models like CLIP or other deep learning frameworks.
    - For combined media: Methods like OCR for text in images or audio-to-text for videos can help to analyze all forms of content.
4. **Classification Model**:
    - A machine learning model, often a neural network or ensemble model, is trained to classify content as "harmful" or "not harmful."
    - This model may use both **predefined rules** (such as specific banned words) and **deep learning models** trained on large datasets labeled for harmful content (e.g., hate speech, misinformation, etc.).
5. **Output Decision**:
    - If the content is classified as harmful, it is either **flagged** for human review, **removed**, or subjected to other penalties like content demotion.
    - If the content is classified as safe, it's allowed to remain on the platform.

- Specifying system's input/output:



- The post can be heterogeneous and can multimodal.
  - Two common modalities:
    - Late Fusion
    - Early Fusion

Late Fusion:

A **late fusion model** is a machine learning or deep learning approach where different types of features (or information from multiple modalities, like text, image, or audio) are processed separately and independently through separate models or sub-networks. Each model learns to extract features from its specific modality, and the outputs of these models are then **combined or "fused" at a later stage**, typically just before the final prediction layer.

Key Characteristics of Late Fusion:

1. **Separate Processing**: Each modality (e.g., text, image, audio) is handled by its own dedicated model or sub-network.
2. **Fusion Layer**: The outputs of these models are combined at a later stage, usually by concatenating, averaging, or performing other operations on the final feature vectors.
3. **Final Prediction**: A single prediction is made based on the combined features.

Example in Practice:

In a social media content moderation system, a late fusion model might separately process:

- **Text** through an NLP model to understand sentiment or detect offensive language.
- **Images** through a CNN to detect potentially harmful or explicit content.
- **Metadata** (like user history or post timestamps) through another model.

The outputs from these individual models are then fused to make the final decision on whether the content is harmful.

Advantages:

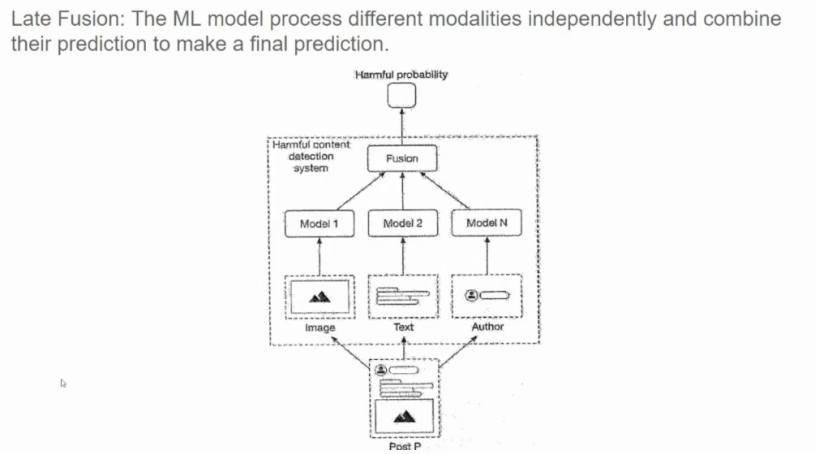
- **Modular**: Each modality can be fine-tuned separately, allowing specialized models to handle different data types effectively.
- **Interpretability**: Since features are extracted separately, it's easier to analyze the contribution of each modality.

Disadvantages:

- **Increased Complexity**: Requires managing multiple models, which can increase computational costs.
- **Information Loss**: Fusion at a later stage may miss interactions between modalities that could be useful for better prediction.

Application Areas:

Late fusion is widely used in tasks like multimedia content analysis, multimodal sentiment analysis, and recommendation systems, where data often comes from diverse sources that can contribute uniquely to the final prediction.



Early Fusion:

An **early fusion model** is a machine learning approach where data from multiple modalities (such as text, image, audio, etc.) is **combined at the input level or in the initial stages of processing**. This fused input data is then fed into a single model that learns to jointly process and analyze the combined features.

Key Characteristics of Early Fusion:

1. **Combined Input**: Different types of data (modalities) are merged early in the model pipeline, often by concatenating raw features or embeddings before feeding them into the model.
2. **Single Model**: A single network learns to process the fused input and make predictions, leveraging interactions between the modalities from the beginning.
3. **Unified Learning**: The model learns relationships between features across modalities jointly, capturing cross-modal interactions more directly.

Example in Practice:

In a social media content moderation system, early fusion could involve:

- Combining **text embeddings** from captions or comments with **image embeddings** from photos in a single vector.
- Feeding this unified vector into a single model to detect harmful content, taking advantage of the relationship between the text and visual information from the start.

Advantages:

- **Cross-Modal Interactions**: Allows the model to capture relationships between modalities early on, which can improve accuracy for tasks where interactions are important.
- **Simplified Pipeline**: Only one model processes the fused data, which can be simpler than managing multiple networks.

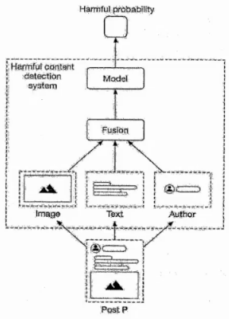
Disadvantages:

- **Data Dependency**: Requires data from all modalities at the input stage, which may not be available in all scenarios.
- **Potential Overfitting**: The model may overfit if modalities vary greatly in information content or quality, especially with smaller datasets.

Application Areas:

Early fusion is useful in applications like multimodal sentiment analysis, video classification, and tasks where multiple data sources need to be understood together, such as detecting sarcasm in text paired with images or identifying harmful content that depends on both text and visual cues.

Early Fusion: The modalities are combined first and then the prediction are made.



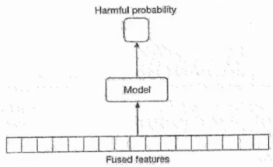
Two common modalities:
- Early Fusion:
  - Pros and Cons:
    - Two major advantages:
      - Same training data for the model.
      - Captures the whole modality i.e. if combination of modalities are harmful the model can potentially capture in the unified feature vector.
    - Disadvantage:
      - Learning the task is difficult for the model due to the complexities involved, in absence of sufficient data is challenging for the model to learn complex relationships and make good predictions.

We will use early fusion - 1. It allows us to capture posts that are harmful overall despite each modality being benign individually, additionally 500 million posts are being published every day, the model had enough data to learn the task.

4. Choosing the Right Model/ Model Selection

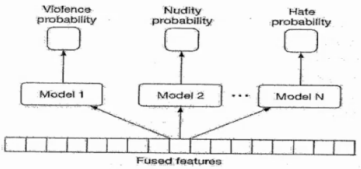
Choosing the right ML category:
- Single Binary Classifier
- One binary classifier per harmful post
- Multi-class label classifier
- Multi-task classifier

1. Single Binary Classifier: The model takes in fused feature as input and predicts the probability of the post being harmful. Since the output is binary it is called binary classifier.



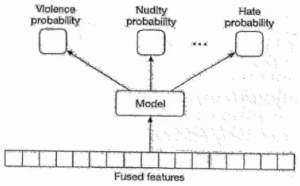
Limitation:
- We get the output of post being harmful but do not know which class of “harm” does that post belongs to.
  - Two issues arise due to this:
    - We cannot say user why the post was removed.
    - If the model is not performing well in some classes it is not possible to know and improve the model on underperforming classes.

2. Binary Classifier per harmful class:Each model predicts the probability of the post being harmful.



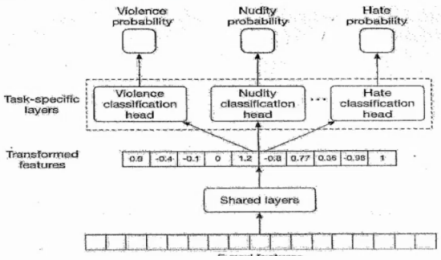
- Each model takes fused features as input and predicts the probability of the post being classified as a harmful class.
- Advantage - We can explain to user why a post was taken down and also evaluate and improve the models independently.
- Disadvantage - As we have multiple models, we have to train them and maintain them separately which is time consuming as well as expensive.

3. Multi-Label Classifier:
- The data point can belong of arbitrary number of classes, so a single model is used as a multi-model classifier.
- Input - fused feature
- Output - probability of harmful class.

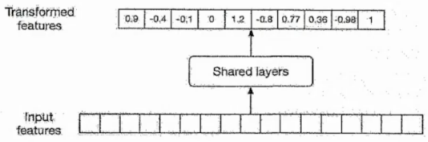


Advantage - training and maintaining the model is less costly.
Disadvantage - predicting the probabilities of each harmful class using a shared model isn't ideal as input features may need to be transformed differently.

Multi-task classifier:
- Multi-class learning - A process in which the model learns multiple tasks simultaneously which allows the model to learn similarities between tasks. We avoid doing unnecessary computations when a certain input transformation is beneficial for multiple tasks.
- We are treating different classes (violence, hate speech, etc) as different tasks and use this model to learn each task.



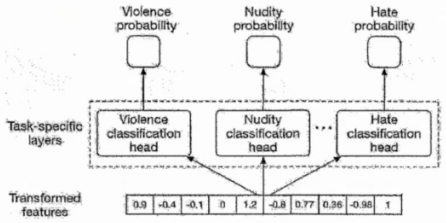
Multi-task Classifier:
1. Shared Layers:
- Transforms input features into new ones which are use to make predictions for each of harmful classes.



Multi-task Classifier:

2. Task Specific Layers:

Task specific layers are independent ML layers, also known as classification heads, which transforms the features in a way which are optimal in predicting the classes.

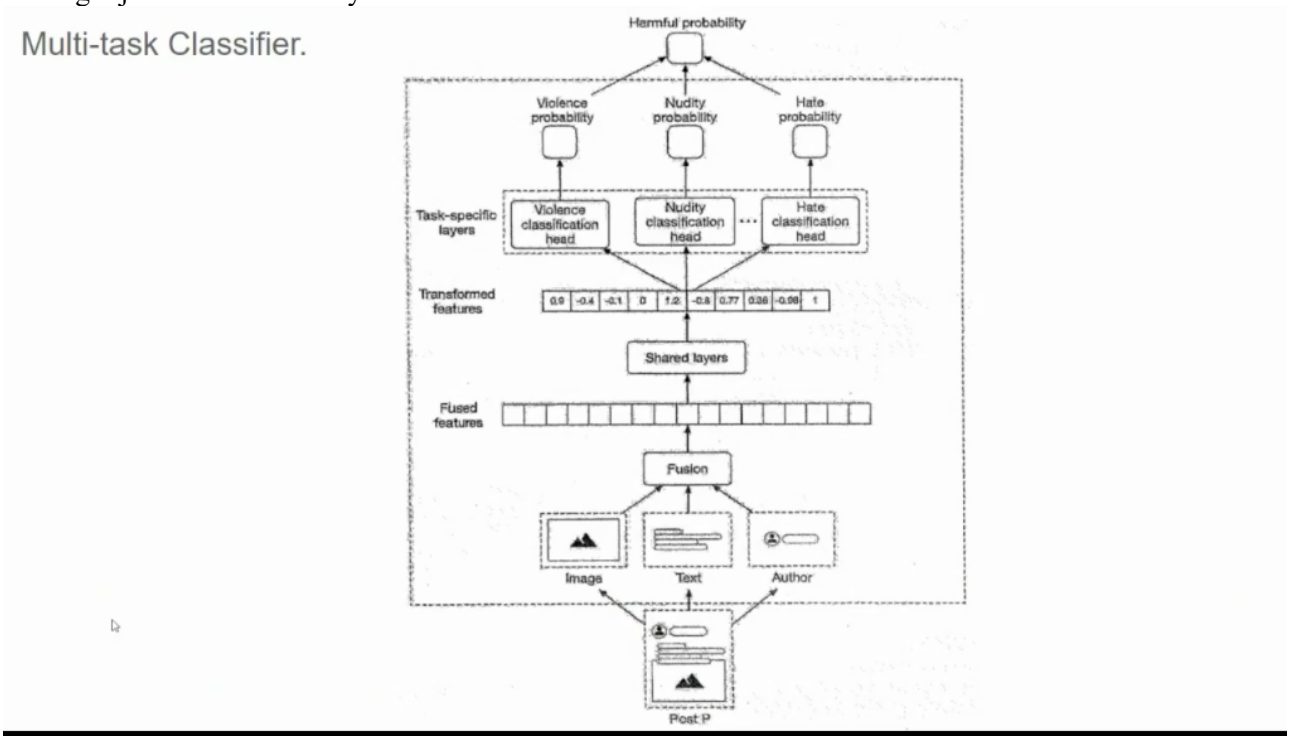


Advantage:

- Using a single model helps to train and maintain it.
- The shared layer transforms the feature in such a way that it is beneficial for each task which reduces the redundancy to compute on each feature for each model.
- Training data for each task contributes to training of other tasks.

For these advantages we will use Multi-task Classifier.

So, the final architecture is to use a multi-task classifier. But it is important to clarify at this point whether the interviewer agrees with this, or this might be too complicated. For example, if we don't have huge labeling budget then we might just start with binary classifier and then later move towards multi-task classifier.



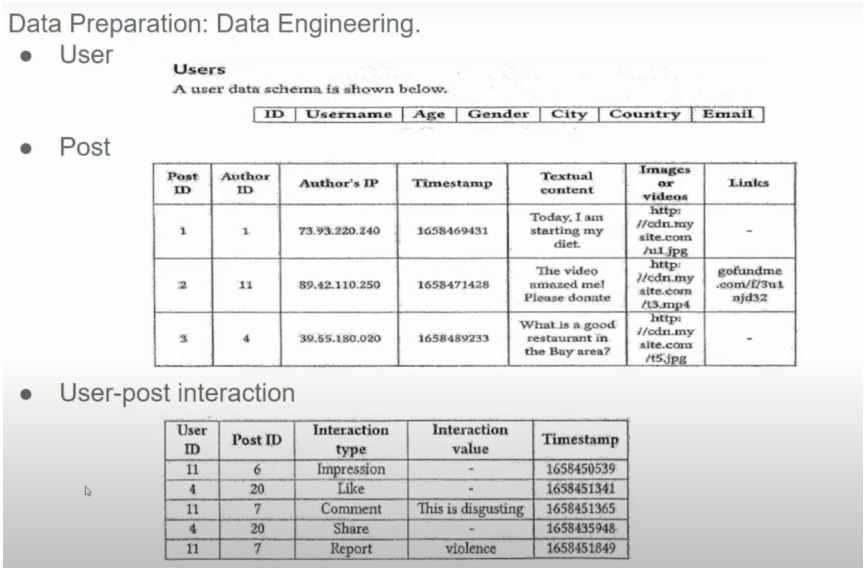
This diagram represents a **multi-task classifier** designed to detect harmful content on social media platforms. Here's a breakdown of the key components:

1. **Input Data (Bottom Section)**:
    - The model receives different types of input data, including **images**, **text** (such as post captions or comments), and **author information** (potentially metadata about the user posting the content).
2. **Fusion Layer**:
    - The inputs from various modalities (image, text, and author) are combined at this fusion layer, producing a **fused feature representation**. This layer likely represents **early fusion**, where features from different modalities are merged early in the model pipeline to create a unified input for further processing.
3. **Shared Layers**:
    - The fused features are then passed through **shared layers**. These layers learn general patterns that apply to all tasks, allowing the model to leverage shared information across different harmful content types.

4. **Task-Specific Layers and Heads**:
    - After the shared layers, the model branches out into **task-specific layers** that focus on different types of harmful content, such as **violence**, **nudity**, and **hate**.
    - Each of these branches ends in a **classification head**, which outputs the probability that the content contains violence, nudity, or hate.
5. **Output Probabilities (Top Section)**:
    - The final output of each classification head is a probability score indicating the likelihood that the content is harmful in each of the defined categories.
    - These probabilities can then be used to make decisions about whether to allow, flag, or block the content.

This multi-task classifier is beneficial for social media platforms as it allows them to detect and categorize potentially harmful content across multiple dimensions (violence, nudity, hate speech) in a single model. By sharing features between tasks and using specific layers for each type of harmful content, the model can efficiently handle a wide range of content moderation needs, helping to keep platforms safe and adhering to content policies. Generally, these classifiers are also fast (which is one of the requirements of such model) since the embeddings are created using a pre-trained model.

6. <u>Feature Engineering</u>



6.1 Text Processing

Data preparation:
Feature Engineering:

- Textual content
- Image or video
- Author
- User reaction to the post
- Contextual information

1. Textual content - to use textual content of the post to predict whether the post is harmful or not.
   a. Text preprocessing (normalization, tokenization etc.)
   b. Vectorization - to convert the text into meaningful vectors.
      i. Statistical Method (Bag of Words - BoW, Term Frequency - Inverse Document Frequency TF-IDF)
      ii. ML Based Method - Transformer based architecture.
         1. BERT - Bidirectional Encoder Representations from Transformers.
            a. It is slow and not ideal for real time prediction.
            b. It was only trained on English. A variant of BERT is DistilmBERT.

Feature	**BERT**	**DistilBERT**
Developed by	Google	Hugging Face
Year of Release	2018	2019
Model Size	Large (e.g., BERT-base has 110M parameters)	Smaller (about 66M parameters, 40% fewer than BERT)
Architecture	Bidirectional Transformer	Similar to BERT but with fewer layers
Layers	12 (BERT-base), 24 (BERT-large)	6 (roughly half of BERT-base)
Hidden Size	768 (BERT-base), 1024 (BERT-large)	768 (same as BERT-base)
Attention Heads	12 (BERT-base), 16 (BERT-large)	12 (same as BERT-base)
Training Objective	Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)	Distilled from BERT, trained on MLM only
Speed	Slower due to larger architecture	About 60% faster inference than BERT
Memory Requirement	High	Lower memory requirement due to reduced size
Accuracy	Slightly higher accuracy on most NLP tasks	Slightly lower accuracy, but close to BERT's
Intended Use	High-resource applications where accuracy is key	Resource-constrained environments needing efficiency
Languages Trained On	Multilingual (trained on 104 languages in multilingual variant)	Primarily English; multilingual versions available through distillation
Model Variants	BERT-base, BERT-large, multilingual BERT, BERT for different languages (e.g., Chinese BERT, Arabic BERT)	DistilBERT-base (English), DistilBERT-multilingual (trained on 104 languages), other distillation adaptations for specific tasks

BERT has a variety of language-specific and multilingual versions, covering over 100 languages in its multilingual variant. **DistilBERT**, originally trained for English, also has a multilingual variant adapted through distillation, and there are other adaptations of DistilBERT for specific languages or tasks. DistilBERT is designed to be faster and more resource-efficient while preserving much of BERT's performance.

6.2 Image or Video

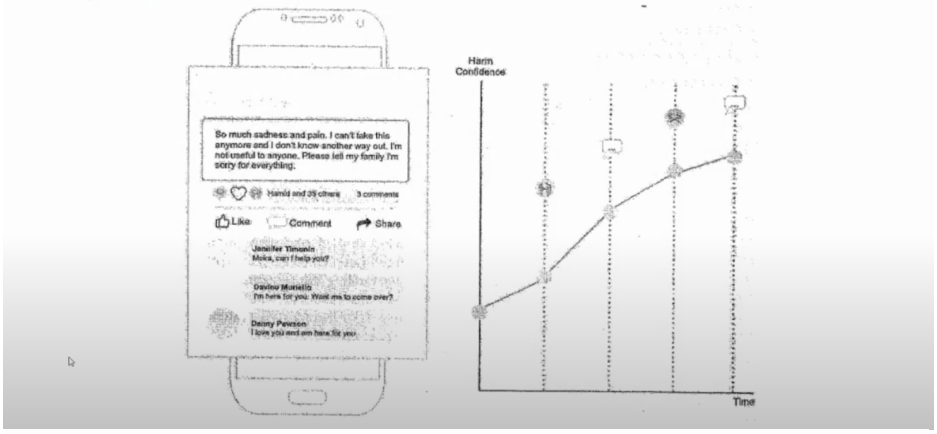
Image or Video:

- To know the contents of the image or video, which uses
  - Preprocessing - decoding, resize, normalize the data.
  - Feature extraction - to convert the unstructured data into feature vector.
    - For image - CLIP's visual encoder or SimCLR (Simple Framework for Contrastive Learning of Representation).
    - For video - VideoMoCo (Momentum Contrast).

6.3 User Post Interaction

Feature Engineering:

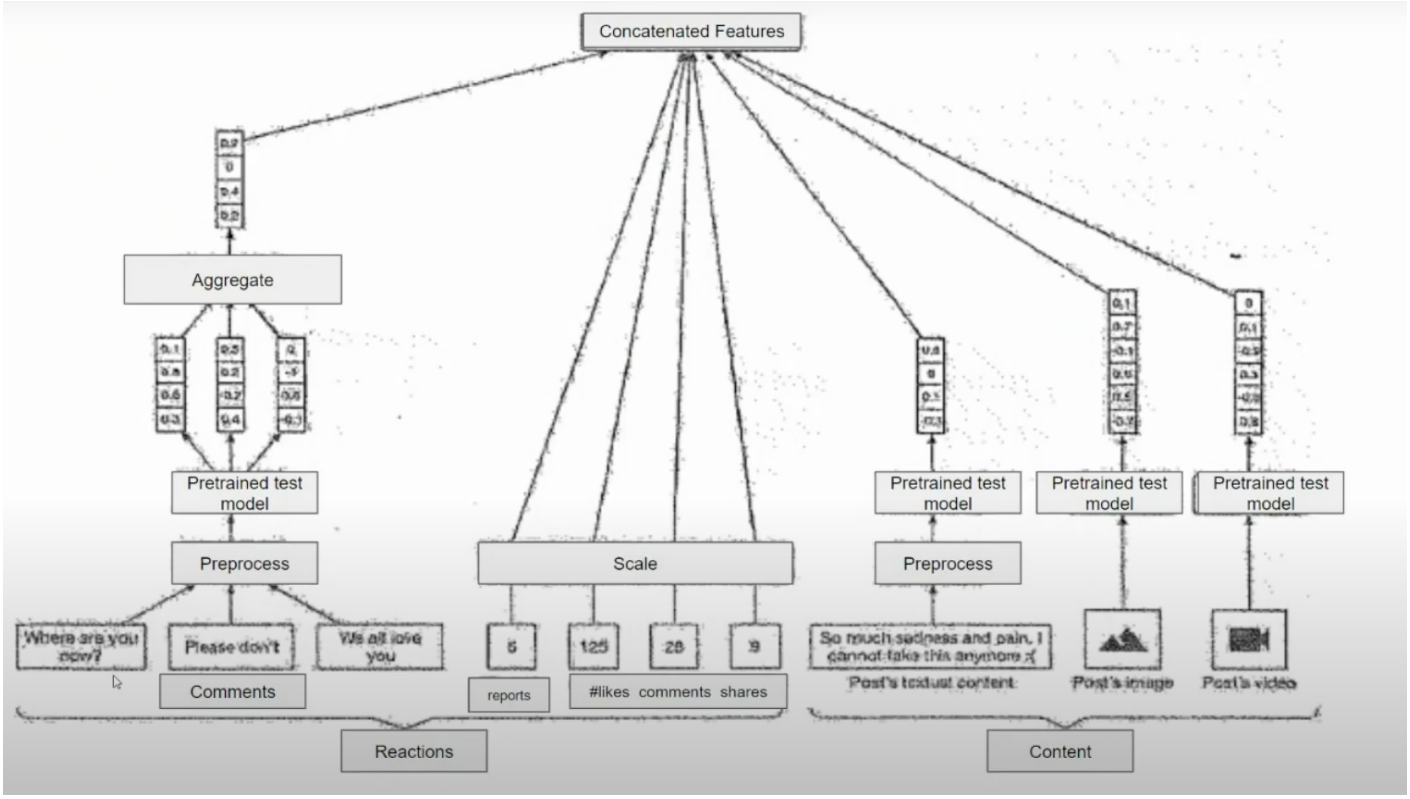
User reactions to the post: helps determine if post is harmful especially when the post is ambiguous.



User post reactions -

1. Number of likes, comments, shares and reports- these values are used to speed up convergence during training process.
2. Comments - comments can help us identify harmful content.
   a. Using comments in Feature engineering:
      i. Using the pretrained model to Convert the comments into numerical representations (embeddings) for each comment.
      ii. Aggregating the embeddings to create a final embedding.

6.4 All features combined



 Input **Modalities**:

- The diagram takes multiple types of input from a social media post:
    - **Comments**: Text from comments associated with the post.
        - The user aggregates the comments rather than concatenates them likely to summarize or capture an overall sentiment or pattern from multiple comments, rather than treating each comment individually. Aggregation (e.g., averaging sentiment scores or combining features) can provide a more cohesive representation of the comments' collective tone, reducing dimensionality and focusing on the overall impact, which is useful for efficient processing and model input.
    - **Reactions**: Engagement metrics like the number of reports, likes, comments, and shares.
        - Since these values can vary widely (e.g., likes could range from tens to thousands, while reports might be fewer), scaling helps to bring them into a similar range. This avoids any single feature, like high counts of likes, from disproportionately influencing the model's predictions. Common scaling techniques include min-max scaling (to a 0–1 range) or z-score normalization. This step is essential to ensure that the model treats each feature equally and can effectively interpret their relative significance.
    - **Content**: The post's own textual content, images, and videos.

 Preprocessing:

- Each input type undergoes preprocessing to prepare it for model analysis. This likely includes:
    - **Text Processing**: Converting text into a format suitable for analysis (e.g., tokenization, vectorization).
    - **Image and Video Processing**: Transforming visual content (like images and videos) into feature representations using computer vision techniques.

 Feature **Extraction Using Pretrained Models**:

- For each input type, a **pretrained model** (e.g., NLP models for text, CNN models for images) is used to extract features. These features capture relevant characteristics such as sentiment, content themes, or visual elements that may signal harmful content.
- This step converts different data types into numerical feature vectors, allowing them to be processed together.

 Scaling **and Aggregating Reaction Features**:

- The reaction data, like the number of likes, comments, shares, and reports, is scaled or normalized. This step helps standardize the features so they can be used effectively in downstream processing.

 Concatenation **of Features**:

- All processed features (from comments, reactions, and content) are concatenated into a single feature vector. This forms a comprehensive representation of the post, capturing information from various modalities.

 Fusion **Layer**:

- After concatenation, the fused feature vector is used as input to a downstream classifier, which could be a multi-task classifier. This classifier can determine the probability of different harmful categories, such as spam, hate speech, nudity, or violence.

6.5 Author's Features

Author Features: author's past interaction to determine the posts is harmful or not.

Author's violation history:

- Number of violations - numerical value of number of times user violated the guidelines in the past.
- Total user reports - numerical value of number of times users reported author's post.
- Profane words rate - numerical value of number of times in the past the user's post or comments contained profane words. A predefined list is used to determine whether a word is profane.

Author's demographics

- Age - User's age is one of the most important predictive features.
- Gender - represents gender of the user. One-hot encoding is used to represent gender.
- City and Country - Both the city and country take many distinct values. To represent features, we use embedding layer to convert city and country into feature vectors. One-hot encoding is should not be used here.

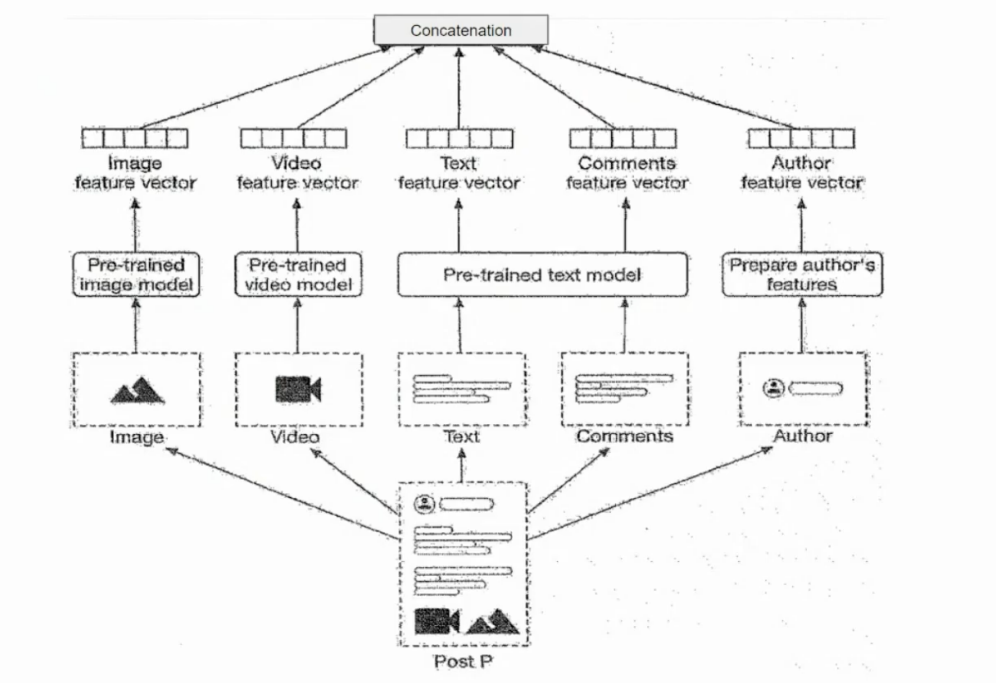
Account Information:

- Number of followers and followings
- Account Age - numerical representation of the age of the author's account. This is a predictive feature as accounts with a lower age are more likely to be spam or to violate integrity.

Contextual Information:

- Time of day - the time of day when author made a post. We bucketize into multiple categories as morning, noon, afternoon, evening or night.
- Device - The device the author uses eg. mobile phone or desktop computer.
  For both, we use One-hot encoding.

6.6 Final Features



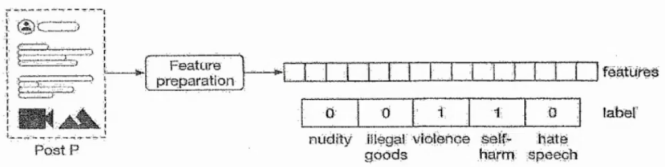
7. Model Development

Model Development:

- We will use neural network for multi-task learning.
- To work with neural network, we have to determine:
  - Neural network's architecture
  - Optimal hyperparameters - hidden layers, activation functions, learning rate etc.
    - To choose optimal hyperparameters, hyperparameter tuning is used which is a process of finding best parameters for getting the best performance from the model.
      - This involves:
        -> Training a model for each combination of hyperparameter values
        -> Evaluating each model
        -> Selecting the hyperparameter that leads to best model .

Model Development:
Model Training:
Constructing the dataset:

- The model's input are features and outputs labels that model is expected to predict.
- Two ways to construct the dataset for input:
  - Hand labelling - where a human contractors are labelling manually.
  - Natural labelling - we use user reports to label posts automatically.
- For Evaluation Dataset - we use hand labelling to prioritize the accuracy of labels.
- For training dataset - we use natural labelling to prioritize labelling speed.



Evaluation Dataset:

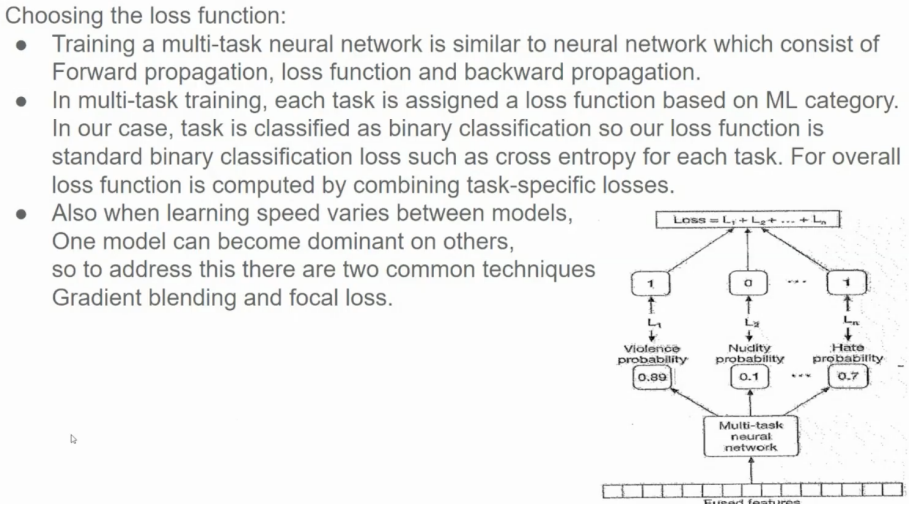
- **Hand labeling:** This method involves human annotators carefully labeling each data point to ensure accuracy. This is often more time-consuming but guarantees high-quality labels.

Training Dataset:

- **Natural labeling:** This approach uses less rigorous labeling methods, such as crowdsourcing or automatic labeling techniques. It prioritizes speed over accuracy, making it suitable for large-scale datasets.

This strategy is common in machine learning, where a small, high-quality evaluation dataset is used to assess the performance of a model, while a larger, less rigorously labeled training dataset is used to train the model.

8. Loss Function



Loss Function: The overall loss of the model is calculated as the sum of the individual losses for each task. This encourages the model to learn representations that are useful for all tasks simultaneously.

Gradient Blending

Gradient blending is a technique used to combine multiple models or their gradients to improve overall performance. This technique is particularly useful when dealing with complex tasks or when different models have complementary strengths.

How it works:

1. **Multiple Models:** Train multiple models on the same dataset, potentially with different architectures or hyperparameters.
2. **Gradient Calculation:** During training, calculate the gradients for each model.
3. **Gradient Blending:** Combine the gradients from different models using a weighted average or other techniques.
4. **Model Update:** Update the model parameters using the blended gradients.

Benefits of Gradient Blending:

- **Improved Performance:** Can lead to better performance by combining the strengths of different models.
- **Reduced Overfitting:** Helps to reduce overfitting by introducing diversity in the training process.
- **Enhanced Generalization:** Can improve the generalization ability of the model to unseen data.

Focal Loss

Focal loss is a loss function designed to address the class imbalance problem in object detection and classification tasks. It focuses on hard examples, giving them more weight in the loss calculation, while downplaying the contribution of easy examples.

How it works:

1. **Cross-Entropy Loss:** It's based on the cross-entropy loss, which measures the dissimilarity between the predicted probability distribution and the true distribution.
2. **Focusing Parameter:** It introduces a focusing parameter (gamma) that modulates the loss based on the difficulty of the example.
3. **Difficulty Modulation:** Easy examples (with high predicted probability) contribute less to the loss, while hard examples (with low predicted probability) contribute more.

Benefits of Focal Loss:

- **Improved Accuracy:** Better handling of class imbalance, leading to more accurate predictions.
- **Faster Convergence:** Faster training by focusing on the most informative examples.
- **Robustness:** More robust to noisy data and outliers.

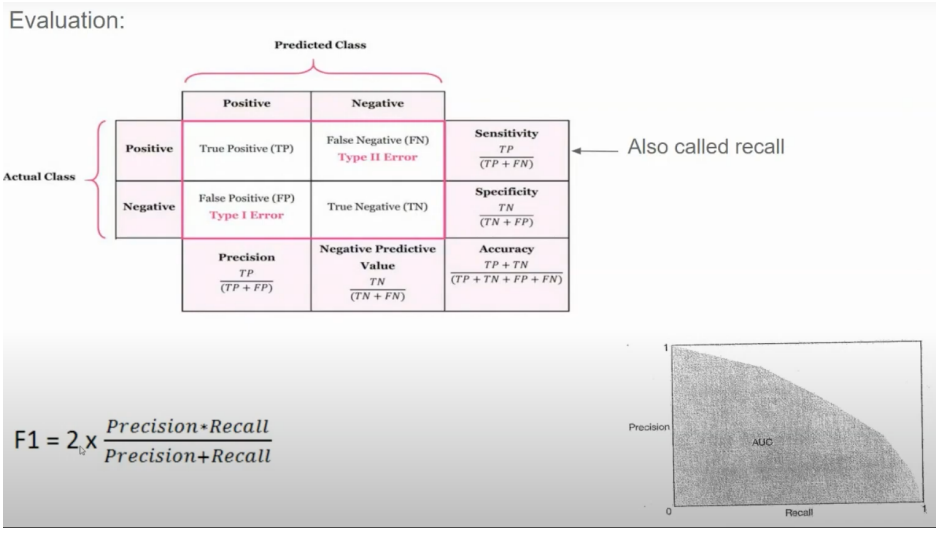
Combining Gradient Blending and Focal Loss:

By combining gradient blending with focal loss, you can further improve the performance of your models:

- **Diverse Models:** Train multiple models with different architectures or hyperparameters.
- **Focal Loss:** Apply focal loss to each model to focus on hard examples.
- **Gradient Blending:** Combine the gradients from these models to obtain a more robust and accurate solution.

This combination can lead to significant improvements in model performance, especially in challenging scenarios with imbalanced datasets and complex tasks.

9. Evaluation

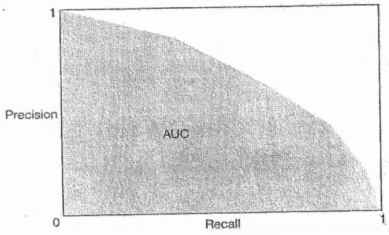


Offline Evaluation:

- Metrics like precision, recall and f1 scores are used.

1. PR- Curve (Precision Recall Curve) - PR curve is obtained by plotting precision at different probability thresholds ranging from 0 to 1.
   PR - AUC (Area Under the Curve) is calculates the area beneath the PR curve.
2. Receiver Operating Characteristic (ROC) curve - similar to PR Curve, shows the tradeoff between true positive rate (recall) and false positive rate. AUC for ROC is calculated.

We will use both PR-Curve and ROC-Curve.



1. **Precision-Recall Curve (PR Curve)**:
   - **Precision:** Measures the proportion of positive predictions that are actually positive.
   - **Recall:** Measures the proportion of actual positive cases that are correctly identified as positive.
   - **PR Curve:** Plots precision against recall for different classification thresholds.
   - **PR-AUC:** Calculates the area under the PR curve, providing a single metric to assess the overall performance. A higher PR-AUC indicates better performance.
2. **Receiver Operating Characteristic (ROC) Curve:**
   - **True Positive Rate (TPR):** Measures the proportion of actual positive cases that are correctly identified as positive (same as recall).
   - **False Positive Rate (FPR):** Measures the proportion of actual negative cases that are incorrectly identified as positive.
   - **ROC Curve:** Plots TPR against FPR for different classification thresholds.
   - **ROC-AUC:** Calculates the area under the ROC curve. A higher ROC-AUC indicates better performance.

Choosing the Right Metric:

The choice between PR-AUC and ROC-AUC depends on the specific use case and the class imbalance in the dataset:

- **Imbalanced Datasets:** In cases where the classes are imbalanced (e.g., fraud detection), PR-AUC is often preferred as it focuses on the positive class.
- **Balanced Datasets:** ROC-AUC is a good choice for balanced datasets, as it considers both true positive rate and false positive rate.

Additional Considerations:

- **Confusion Matrix:** A confusion matrix provides a detailed breakdown of correct and incorrect predictions, helping to identify specific areas where the model might be struggling.
- **F1-Score:** The F1-score is a harmonic mean of precision and recall, providing a balanced metric.
- **Calibration:** It's important to ensure that the model's predicted probabilities accurately reflect the true probabilities of the positive class. Calibration techniques can be used to improve the model's confidence scores.

By carefully evaluating these metrics and considering the specific context of the problem, you can gain valuable insights into your model's performance and make informed decisions about its deployment.

PR-AUC (Precision-Recall Area Under Curve) and **ROC-AUC (Receiver Operating Characteristic Area Under Curve)** are both metrics used to evaluate the performance of binary classifiers, but they highlight different aspects of the classifier's performance, especially in the context of **class imbalance** (where one class is much more common than the other). Let's break down each one in detail, discuss their differences, and explain when each should be used.

---

1. ROC-AUC (Receiver Operating Characteristic Area Under Curve)

Explanation

- **ROC Curve**: The ROC curve plots the **True Positive Rate (TPR)** (or Recall) on the y-axis against the **False Positive Rate (FPR)** on the x-axis, across different thresholds for classification.
    - **True Positive Rate (TPR)**, also known as **Recall**, is the ratio of correctly identified positive cases to the actual positives: TPR=True Positives (TP)True Positives (TP)+False Negatives (FN)\text{TPR} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}TPR=True Positives (TP)+False Negatives (FN)True Positives (TP)
    - **False Positive Rate (FPR)** is the ratio of negative cases incorrectly classified as positive to the actual negatives: FPR=False Positives (FP)False Positives (FP)+True Negatives (TN)\text{FPR} = \frac{\text{False Positives (FP)}}{\text{False Positives (FP)} + \text{True Negatives (TN)}}FPR=False Positives (FP)+True Negatives (TN)False Positives (FP)
- **ROC-AUC**: The area under this curve, **ROC-AUC**, quantifies the classifier's ability to discriminate between positive and negative classes. An ROC-AUC score of 1 indicates perfect classification, while a score of 0.5 suggests no discrimination ability (random guessing).

Interpretation

- The ROC-AUC is a good measure of **how well the model separates the two classes** by adjusting thresholds. It works well if the **class distribution is relatively balanced**.
- High FPR doesn't affect ROC-AUC as much as it affects PR-AUC, so the ROC-AUC can remain high even if there are many false positives, which may mask poor performance on the positive (minority) class.

Example

Suppose we're building a classifier to detect spam emails (positive class = spam, negative class = non-spam). If we have 1,000 emails, with 500 spam and 500 non-spam, ROC-AUC will give a balanced view of how well our model can separate spam from non-spam emails.

---

2. PR-AUC (Precision-Recall Area Under Curve)

Explanation

- **Precision-Recall Curve**: The PR curve plots **Precision** on the y-axis against **Recall (TPR)** on the x-axis, across different classification thresholds.
    - **Precision** is the ratio of correctly predicted positive cases to all predicted positives: Precision=True Positives (TP)True Positives (TP)+False Positives (FP)\text{Precision} =

\frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}Precision=True Positives (TP)+False Positives (FP)True Positives (TP)

    - **Recall** is the same as in ROC, defined as the ratio of correctly identified positives to all actual positives.

- **PR-AUC**: The area under the PR curve, **PR-AUC**, gives a measure of the classifier's performance on the positive (minority) class, focusing specifically on the **balance between Precision and Recall**.

Interpretation

- PR-AUC is more informative when dealing with **highly imbalanced data** where the positive class (e.g., spam) is rare compared to the negative class (e.g., non-spam).
- PR-AUC emphasizes performance on the minority class, so it better reflects a classifier's ability to make accurate positive predictions when false positives are significant.

Example

Returning to our email example, suppose now that out of 10,000 emails, only 500 are spam, and 9,500 are non-spam. In this case, PR-AUC becomes more informative because the model's performance on the minority class (spam) is critical, and we want to ensure we don't label too many non-spam emails as spam.

Key Differences Between ROC-AUC and PR-AUC

Metric	**ROC-AUC**	**PR-AUC**
Curve	Plots TPR (Recall) vs. FPR	Plots Precision vs. Recall
Emphasis	Measures overall performance, balancing both classes	Focuses on the positive class performance (Precision)
Good for	Balanced datasets or when negatives are equally important	Imbalanced datasets where positives are more important
Insensitive to	Class imbalance (can be misleading in imbalanced datasets)	Class imbalance (highlights performance on minority class)
Low false positives effect	Often tolerated due to FPR in denominator	Can significantly impact Precision and hence PR-AUC
Example Usage	Disease detection in general population	Fraud detection where fraud cases are rare

---

When to Use Each

1. **Use ROC-AUC when**:
    - The **dataset is balanced** or nearly balanced between the classes.
    - **True negatives** are as important as true positives (i.e., misclassifying the negative class has significant consequences).
    - You want an overall measure of performance that considers both positive and negative classifications equally.

2. **Use PR-AUC when**:
    - The **dataset is highly imbalanced**, with one class (often the positive class) being rare.
    - **False positives** are more problematic than false negatives (e.g., marking non-spam emails as spam).
    - You're primarily interested in the model's ability to **identify positives accurately** and ensure that the positive predictions are indeed correct (high precision).

Example Scenario Comparison

Let's look at a credit card fraud detection system, where fraud cases are rare.

- **ROC-AUC**: Suppose the model achieves an ROC-AUC of 0.95. This seems very high, suggesting the model is good at separating fraud from non-fraud cases. However, since the ROC curve does not highlight the model's performance on the minority class, it might mask the fact that the model has many false positives (non-fraud transactions flagged as fraud) or false negatives (fraudulent transactions missed).
- **PR-AUC**: In the same fraud detection model, if we look at PR-AUC, we might find that the PR-AUC is lower (e.g., 0.60), reflecting the difficulty of identifying actual fraud cases without many false positives. The PR-AUC more accurately shows the model's ability to identify fraud cases, making it more suitable for this highly imbalanced dataset.

In summary:

- **ROC-AUC** is better for **balanced datasets** or when you care equally about both classes.
- **PR-AUC** is better for **imbalanced datasets**, especially when accurate detection of the positive class (minority class) is more critical than overall performance across both classes.

Online Metric:

1. Prevalence

$$\text{Prevalence} = \frac{\text{Number of harmful posts we didn't prevent}}{\text{Total number of posts on the platform}}$$

- It treats a post with 100k views same as 10 views, whereas post with 100k views is more harmful.

2. Harmful Impressions - unlike prevalence, this metric captures how many users got affected by the post using harmful impressions.
3. Valid Appeals

$$\text{Appeals} = \frac{\text{Number of reversed appeals}}{\text{Number of harmful posts detected by the system}}$$

- Percentage of posts that were deemed harmful by the system but were reversed.

4. Proactive Rate -

$$\text{Proactive rate} = \frac{\text{Number of harmful posts detected by the system}}{\text{Number of harmful posts detected by the system} + \text{reported by users}}$$

- Percentage of harmful post deleted by the system before users report it.

5. User report per harmful class - measures the systems' performance by looking into user reports for each harmful class.

Prevalence:

- This metric measures the proportion of harmful posts that the system failed to prevent.
- It's calculated as the ratio of harmful posts that were not detected to the total number of posts on the platform.
- A key limitation of this metric is that it treats all posts equally, regardless of their potential impact. A post with 100,000 views is considered equally harmful as a post with 10 views, which might not accurately reflect the real-world impact.

 Harmful Impressions:

- This metric addresses the limitation of prevalence by considering the number of users who were exposed to harmful content.
- It calculates the total number of views or impressions of harmful posts.
- This metric provides a more accurate measure of the actual harm caused by the content.

 Valid Appeals:

- This metric evaluates the accuracy of the system's decisions by measuring the rate of incorrect detections.
- It calculates the proportion of harmful posts that were flagged by the system but were later overturned by human review.
- A high valid appeal rate indicates that the system is prone to false positives.

 Proactive Rate:

- This metric measures the system's ability to proactively detect harmful content before it's reported by users.
- It calculates the proportion of harmful posts that were detected by the system before being reported by users.
- A high proactive rate indicates that the system is effective in preventing harmful content from reaching users.

 User Report per Harmful Class:

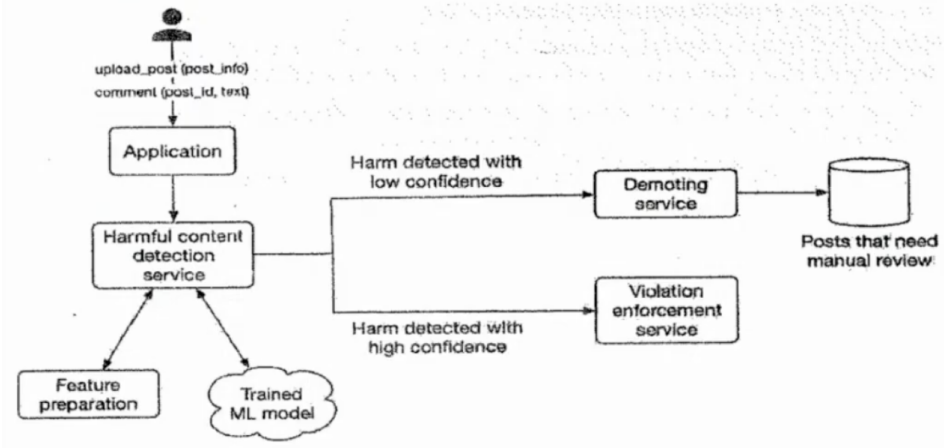
- This metric analyzes user reports for each specific type of harmful content (e.g., hate speech, harassment, etc.).
- It helps identify areas where the system may be underperforming in detecting certain types of harmful content.

Other Metrics

- **False Positive Rate:** Measure the percentage of harmless content that is mistakenly flagged as harmful.
- **False Negative Rate:** Measure the percentage of harmful content that is missed by the system.
- **Processing Time:** Track the time it takes for the system to process and classify content.
- **System Uptime:** Measure the system's availability and reliability.

10. Deployment

Serving:



- Harmful Content Detection Service - This service predicts the probability of harm and some types of harm should be dealt with immediately due to their sensitivity.
- Violating Enforcement Service - It removes the post if Harmful content detection service predicts the harm with high confidence also notifying the user why the post was removed.

1. Application:

- Users can upload posts and comments.

2. Harmful Content Detection Service:

- This service analyzes the uploaded content to determine the probability of it being harmful.
- It can classify content into two categories:
    - **Low Confidence:** Posts that are potentially harmful but require further review.
    - **High Confidence:** Posts that are clearly harmful and should be removed immediately.

3. Demoting Service:

- Handles posts flagged with low confidence.
- It may demote these posts in search results or reduce their visibility to mitigate potential harm.

4. Violating Enforcement Service:

- Removes posts flagged with high confidence.
- Notifies the user about the removal and the reason for it.

5. Posts that Need Manual Review:

- Posts flagged with low confidence are sent to a queue for manual review by human moderators.
- Human moderators can confirm or deny the automated detection and take appropriate action.

6. Feature Preparation and Trained ML Model:

- The system likely uses machine learning models trained on a dataset of labeled harmful and non-harmful content.
- These models extract features from the input content (text, images, etc.) and make predictions about the content's potential harmfulness.

Key Points:

- **Automated Detection:** The system leverages machine learning models to automatically identify harmful content.
- **Human-in-the-Loop:** Human moderators play a crucial role in reviewing and verifying the system's decisions.
- **Adaptive Approach:** The system can adapt to new types of harmful content by retraining the machine learning models.
- **User Notification:** Users are informed about the removal of their content and the reason for it.

- Demoting Service - When the harmful content detection service predicts a post of low harm confidence, the post is demoted to decrease the chance of spreading and then sent to humans for manual assessment and are labelled with predefined classes. These posts are stored for future training iterations to improve the model.

Other talking points:

- Handle Biases introduced by human labelling.
- How to detect authentic and fake accounts.
- How to select post sample for human review.
- How to make harmful content detection system efficient, so we can deploy it on-device.
- How to deal with borderline contents i.e. type of content that are not prohibited by the guidelines but comes close to red lines drawn by those policies.