Executive Summary: Initial Model Development

Project Title:
Automating Comment Handling in Collaborative Documents using Hierarchical Capsule Networks and Advanced AI Techniques

Team Members:
- Sravani Pati
- Vamsi Krishna Pasam

Project Overview:
This project aims to develop a system that automates the handling of comments in collaborative documents. The system will prioritize, categorize, and triage comments based on relevance, complexity, urgency, and importance using advanced machine learning models, including Hierarchical Capsule Networks and Large Language Models (LLMs). The final output will be a comprehensive tool ready for real-world application and publication at the ACM CHI conference.

Phase 1: Data Preparation and Initial Model Development

Objectives:
- To clean and preprocess the data to ensure it is suitable for model training.
- To develop and evaluate initial models using transformer architectures for multi-label classification of comments.

Key Activities:

1. Data Cleaning and Preprocessing:
- **Standardization:** Data formats were standardized, and missing values were handled by filling in 'Unknown' where necessary.
- **Data Splitting:** Comment date and time were split into separate columns, and it was ensured that no empty comments existed.
- **Visualization:** Data visualization techniques were employed to identify critical predictors and trends within the dataset.

2. Initial Model Development:
- **Model Selection:** Various transformer models, including BERT, BART, and RoBERTa, were selected for initial intent classification.
- **Label Expansion:** Only level 0 and level 1 labels were previously defined in the models. Now, the focus has expanded to include all levels (level 0 to level 4), ensuring a more comprehensive categorization and prioritization system.
- **Training:** These models were trained on the cleaned dataset using multi-label classification techniques.
- **Evaluation:** Models were evaluated based on accuracy, precision, recall, F1 score, and mean squared error. The performance metrics were analyzed to identify areas for improvement.

- **Model Inference:** Functions for predicting new text data using the trained models were implemented and tested.

Achievements:

- **Data Preparation:** Successfully cleaned and preprocessed the dataset, ensuring it was ready for model training.
- **Model Training and Evaluation:**
  - XML CNN, BERT, BART, and RoBERTa models were implemented and trained.
  - Achieved promising results with performance metrics indicating a solid foundation for further development.
  - Models could effectively predict multiple labels for comments, covering all levels from 0 to 4.
- **Model Saving and Inference:** Trained models and tokenizers were saved for future use, and inference functions were tested to ensure models' applicability to new data.

Key Metrics:

- BERT Model:

```
        print(f"Loss: {total_loss / len(data_loader):.4f}")
        print(f"Accuracy: {accuracy:.4f}")
        print(f"Precision: {precision:.4f}")
        print(f"Recall: {recall:.4f}")
        print(f"F1 Score: {f1:.4f}")
        print(f"Mean Squared Error: {mse:.4f}")
        print(f"Test Error Rate: {test_error_rate:.4f}")
        print("Confusion Matrix:\n", conf_matrix)

    # Criterion for loss computation
    criterion = torch.nn.BCEWithLogitsLoss()

    # Evaluate the model
    evaluate_model(model, test_loader, criterion, device)
```

```
Loss: 0.2416
Accuracy: 0.3634
Precision: 0.7368
Recall: 0.6801
F1 Score: 0.7073
Mean Squared Error: 0.0721
Test Error Rate: 0.6366
```

- BERT Model Confusion Matrix:

```
Confusion Matrix:
 [[  0   0   0   0   0   0   0   1   0   0   0   0   0   1   0   0   1   0
    3   0]
 [  0  89   3   5  17   8   0   0   0   0   4   0   0  10   0   0   2   0
    0   0]
 [  0   8  67   1  11   6   0   0   0   0   2   0   0  11   2   0   0   0
    0   1]
 [  0  18   2  21  22   8   0   0   1   0   9   0   0   2   0   0   1   0
    0   0]
 [  0   9   2   1  50   7   0   2   1   1   1   1   0   5   1   2   0   0
    1   6]
 [  0  23   7   6  39 164   0   0   1   0  13   0   0  33   4   1   1   0
    0   3]
 [  0   1   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   0   1   0   3   1   0   6   0   0   0   0   0   0   0   2   2   0
    1   1]
 [  0   3   1   2   7   4   0   0  42   0   1   0   0   0   1   1   0   0
    0   0]
 [  0   0   0   0   1   0   0   0   2   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   0   1   0   8   0   0   2   0   0   0   1   0   0   0   4   1   1
    0   0]
 [  0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   5  18   0   8  27   0   0   0   0   1   0   0  45   0   0   0   0
    0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   1   1   0   1   5   0   0   0   1   0   0   0   2   1   0  55   0
    0   1]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]]
```

- XML CNN Model:

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def evaluate_model(model, data_loader):
    model.eval()
    true_labels = []
    predictions = []

    with torch.no_grad():
        for batch in data_loader:
            outputs = model(batch['input_ids'])
            predicted_labels = torch.sigmoid(outputs) > 0.5
            true_labels.append(batch['labels'].cpu())
            predictions.append(predicted_labels.cpu())

    true_labels = torch.cat(true_labels, dim=0).numpy()
    predictions = torch.cat(predictions, dim=0).numpy()

    accuracy = accuracy_score(true_labels, predictions)
    precision = precision_score(true_labels, predictions, average='micro')
    recall = recall_score(true_labels, predictions, average='micro')
    f1 = f1_score(true_labels, predictions, average='micro')

    print(f"Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}, F1 Score: {f1}")

# Evaluate the model
evaluate_model(model, test_loader)
```

```
Accuracy: 0.28428428428428426, Precision: 0.7252695733708392, Recall: 0.65634280865507, F1 Score: 0.6890868596881959
```

- XML CNN Confusion Matrix:

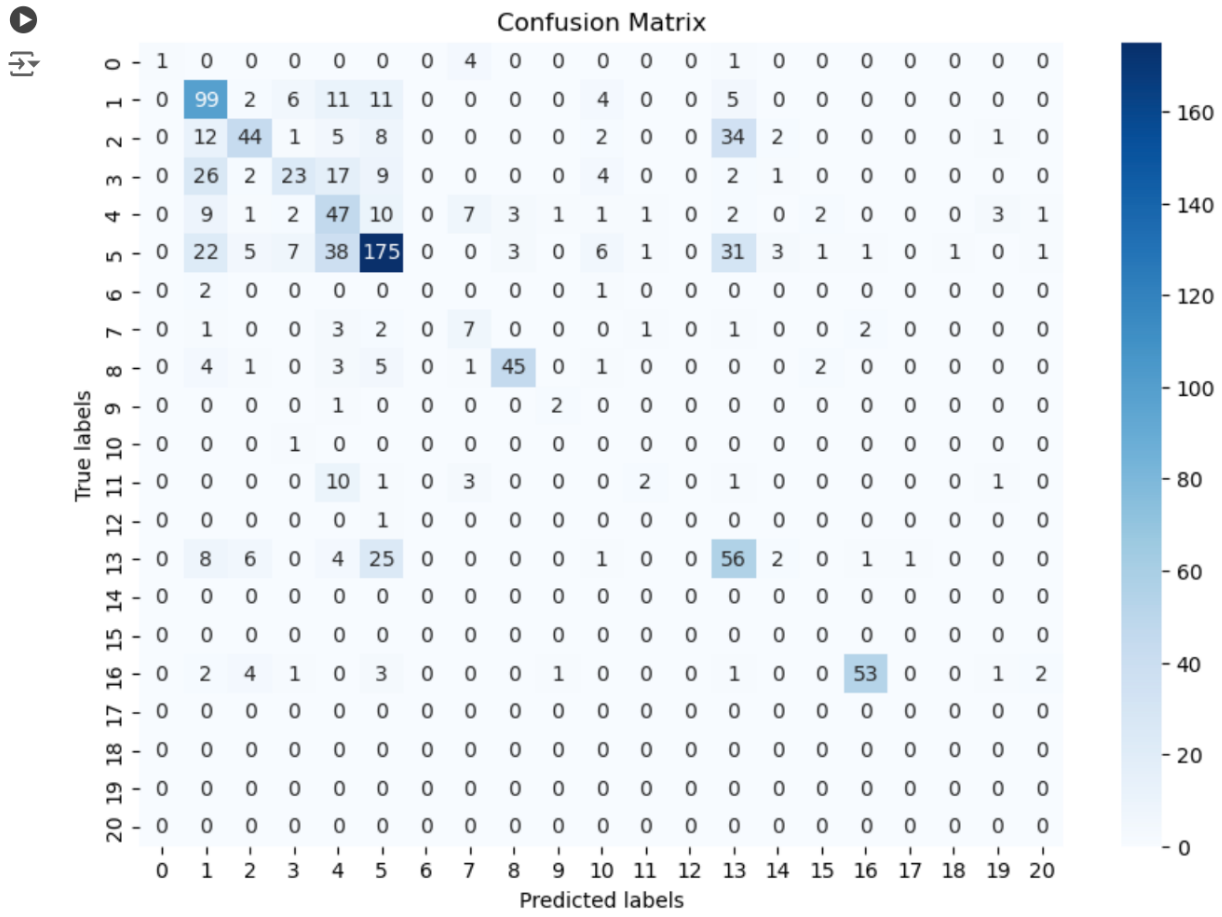
```
Confusion Matrix:
[[  2   0   0   0   0   0   0   1   0   0   0   0   0   1   0   1   0   0
    0   0   1   0]
 [  2  74   4   8   9  17   0   0   0   0   4   0   1   9   0   1   0   0
    0   1   0   8]
 [  0   3  59   3   1   8   0   0   0   0   0   0   0  28   2   1   2   0
    0   1   0   1]
 [  0  21   4  29   6  11   0   0   0   0   7   0   0   3   0   0   0   0
    0   1   0   2]
 [  0   9   2   8  21  10   1   6   0   1   1   7   0   8   1   6   1   0
    0   0   1   7]
 [  1  20   9  13  17 158   0   0   1   2   4   1   0  55   0   0   0   0
    2   3   0   9]
 [  0   0   0   1   0   1   0   0   0   0   0   0   0   0   1   0   0   0
    0   0   0   0]
 [  0   0   1   0   3   1   0   7   0   0   0   0   0   1   0   2   0   1
    0   0   0   1]
 [  0   2   1   4   4   3   0   1  36   0   0   0   0   4   5   0   0   0
    0   0   0   2]
 [  0   0   0   0   0   0   0   0   0   2   0   1   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   0   0   4   0   0   2   0   0   0   4   0   0   0   3   1   0
    0   0   0   4]
 [  0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   5  14   0   3  24   0   1   0   0   2   0   0  48   1   0   0   0
    1   1   0   4]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   4   0   1   2   0   0   0   1   0   0   0   6   0   0  47   0
    0   0   0   7]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0]]
```

- BART Model:

```
        # Print Metrics
        print(f'Loss: {total_loss / len(data_loader):.4f}')
        print(f'Accuracy: {accuracy:.4f}')
        print(f'Precision: {precision:.4f}')
        print(f'Recall: {recall:.4f}')
        print(f'F1 Score: {f1:.4f}')
        print(f'Mean Squared Error: {mse:.4f}')
        print(f'Test Error Rate: {test_error_rate:.4f}')

        # Plot Confusion Matrix
        plt.figure(figsize=(10, 7))
        sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')
        plt.xlabel('Predicted labels')
        plt.ylabel('True labels')
        plt.title('Confusion Matrix')
        plt.show()

# Criterion for calculating the loss
criterion = torch.nn.BCEWithLogitsLoss()

# Example call to evaluate the model
evaluate_model(model, test_loader, criterion, device)
```

```
Loss: 0.2474
Accuracy: 0.3934
Precision: 0.7319
Recall: 0.6922
F1 Score: 0.7115
Mean Squared Error: 0.0725
Test Error Rate: 0.6066
```

- BART Model Confusion Matrix:

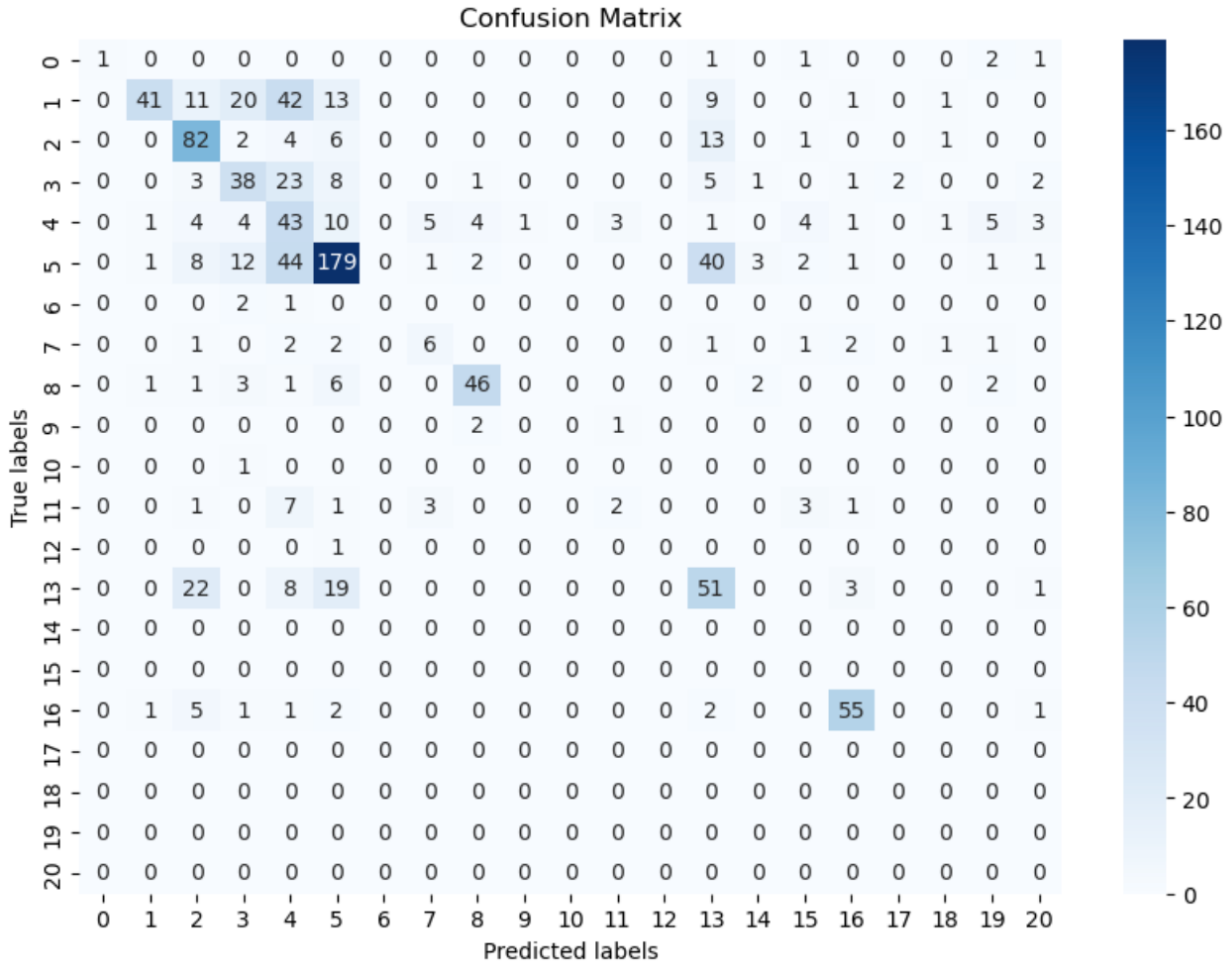


- Roberta Model:

```
        # Output the metrics
        print(f'Accuracy: {accuracy:.4f}')
        print(f'Precision: {precision:.4f}')
        print(f'Recall: {recall:.4f}')
        print(f'F1 Score: {f1:.4f}')
        print(f'Mean Squared Error: {mse:.4f}')
        print(f'Test Error Rate: {test_error_rate:.4f}')

        # Plotting the confusion matrix
        plt.figure(figsize=(10, 7))
        sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
        plt.xlabel('Predicted labels')
        plt.ylabel('True labels')
        plt.title('Confusion Matrix')
        plt.show()

    # Example call to evaluate the model
    evaluate_model(model, test_loader, torch.device('cuda' if torch.cuda.is_available() else 'cpu'))
```

```
Accuracy: 0.3724
Precision: 0.7487
Recall: 0.6871
F1 Score: 0.7166
Mean Squared Error: 0.0711
Test Error Rate: 0.6276
```

- Roberta Model Confusion Matrix:



Conclusion:

The initial model development phase established a strong foundation for automating comment handling in collaborative documents. With data thoroughly prepared and initial models performing well, the project is well-positioned to advance into Phase 2. This phase will focus on developing more sophisticated models, including hierarchical capsule networks and exploring zero-shot and n-shot learning techniques.

Including all levels (0 to 4) in the model, the development ensures a more comprehensive and effective system for categorizing and prioritizing comments. This improvement significantly enhances the project's potential and readiness for real-world application and academic publication.

Next Steps:

- Develop and implement hierarchical capsule networks.
- Explore zero-shot and n-shot learning with LLMs and VLMs.
- Further, refine and integrate models for enhanced performance and usability.

Our team is committed to continuing this momentum, ensuring successful project completion and readiness for real-world application and academic publication.