1. Transformer Model

Dataset Analysis and Preprocessing

The text dataset exhibited significant variance in length distribution:

- Word count range: 13 to 54 words for most entries
- Mean word count: approximately 50 words
- Maximum length: 783 words
- Vocabulary size: approximately 92,000 unique words
- Distribution showed considerable skewness due to outlier entries with exceptionally high word counts
- Standard deviation was notably large, indicating substantial variation in text lengths

Preprocessing Steps:

1. Text normalization:
    - Conversion to lowercase
    - Removal of non-alphabetic characters
    - Reduction of multiple spaces
2. Linguistic processing:
    - Lemmatization using spaCy
    - Stop word removal
3. Outlier handling:
    - IQR-based outlier removal

Post-preprocessing Statistics:

- Typical word count range: 7 to 28 words
- Mean word count: approximately 17.36 words
- Standard deviation: 14.94
- Maximum length reduced to 391 words
- 25th-75th percentile range: 6-23 words

Vectorization and Model Configuration

Vectorization Trials:

- Multiple combinations of token limits and vector dimensions were tested.
- Optimal configuration: 5,000 tokens with 30-dimensional vectors.

Model Details:

- Vocabulary size: 5,000 tokens
- Sequence length: 30 tokens
- Embedding dimension: 100
- Two transformer encoder blocks:
  - Intermediate dimension: 256
  - Number of attention heads: 4
  - Dropout rate: 0.4
- Dense layers:
  - First dense layer: 128 units with ReLU activation
  - Second dense layer: 64 units with ReLU activation
  - Output layer: 5 units with softmax activation
- Regularization:
  - L2 regularization (0.01)
  - Dropout rate: 0.5

Training Configuration

- Optimizer: AdamW with 1e-4 learning rate
- Loss function: Sparse categorical crossentropy
- Batch size: 128
- Early stopping with 5 epochs patience
- Learning rate reduction on plateau

Performance

- Final validation accuracy: 67%
- Training process revealed initial overfitting.
- Regularization and learning rate adjustments improved generalization.

---

2. LSTM Architecture

Dataset Analysis and Preprocessing

Preprocessing Steps:

1. Data scaling:
   - Features were scaled using `MinMaxScaler` or `StandardScaler` to normalize data for better convergence.
   - Example:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

2. Sequence preparation:
   - Data was reshaped into sequences (3D tensors: `(samples, time_steps, features)`) to fit RNN input requirements. Example:

```
import numpy as np
time_steps = 50
X_train_seq = []
y_train_seq = []

for i in range(len(X_train_scaled) - time_steps):
    X_train_seq.append(X_train_scaled[i:i + time_steps])
    y_train_seq.append(y_train_scaled[i + time_steps])

X_train_seq = np.array(X_train_seq)
y_train_seq = np.array(y_train_seq)
```

3. Label encoding (if classification):
   - Labels were one-hot encoded for classification tasks. Example:

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
y_train_encoded = encoder.fit_transform(y_train.reshape(-1, 1)).toarray()
```

4. Batch preparation (optional):

    - Data generators were used for efficient batch preparation. Example:

```
from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
generator = TimeseriesGenerator(X_train, y_train, length=50, batch_size=32)
```

Model Details:

- Embedding layer: 100-dimensional embedding
- LSTM layer: 128 units
- Dense layers:
    - First dense layer: 64 units with ReLU activation
    - Output layer: 5 units with softmax activation
- Regularization:
    - L2 regularization (0.01)
    - Dropout rate: 0.5

Training Configuration

- Optimizer: Adam with 1e-3 learning rate
- Loss function: Sparse categorical crossentropy
- Batch size: 64
- Early stopping with 5 epochs patience

Performance

- Final validation accuracy: 64.6%
- Demonstrated good handling of sequential patterns.
- Balanced performance between training and validation.

3. GRU Architecture

Dataset Analysis and Preprocessing

Preprocessing Steps:

1. Data scaling:

    - Features were scaled using `MinMaxScaler` or `StandardScaler` to normalize data for better convergence.
    - Example:

    ```
    from sklearn.preprocessing import MinMaxScaler
    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    ```

2. Sequence preparation:

    - Data was reshaped into sequences (3D tensors: `(samples, time_steps, features)`) to fit RNN input requirements. Example:

    ```
    import numpy as np
    time_steps = 50
    X_train_seq = []
    y_train_seq = []

    for i in range(len(X_train_scaled) - time_steps):
        X_train_seq.append(X_train_scaled[i:i + time_steps])
        y_train_seq.append(y_train_scaled[i + time_steps])

    X_train_seq = np.array(X_train_seq)
    y_train_seq = np.array(y_train_seq)
    ```

3. Label encoding (if classification):

    - Labels were one-hot encoded for classification tasks. Example:

    ```
    from sklearn.preprocessing import OneHotEncoder
    encoder = OneHotEncoder()
    y_train_encoded = encoder.fit_transform(y_train.reshape(-1, 1)).toarray()
    ```

4. Batch preparation (optional):

- Data generators were used for efficient batch preparation. Example:

```
from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
generator = TimeseriesGenerator(X_train, y_train, length=50, batch_size=32)
```

Model Details:

- Embedding layer: 100-dimensional embedding
- GRU layer: 128 units
- Dense layers:
  - First dense layer: 64 units with ReLU activation
  - Output layer: 5 units with softmax activation
- Regularization:
  - L2 regularization (0.01)
  - Dropout rate: 0.5

Training Configuration

- Optimizer: Adam with 1e-3 learning rate
- Loss function: Sparse categorical crossentropy
- Batch size: 64
- Early stopping with 5 epochs patience

Performance

- Final validation accuracy: 64.2%
- Faster training compared to LSTM.
- Minimal performance trade-off for reduced complexity.

Performance Summary

Model	Validation Accuracy	Key Strengths
Transformer	67.0%	Best contextual understanding
LSTM	64.6%	Strong sequential pattern recognition
GRU	64.2%	Efficient training and simplicity