Model Architectures

1)Classification Model Architecture

The model follows a sequential architecture:

1. An **Embedding layer** (input dim: vocabulary size, output dim: 256, input shape: `(max_sequence_length,)`) is used to convert token indices into dense vector representations. This is followed by a **Bidirectional LSTM layer** with 128 units and a 0.3 dropout rate to capture context from both directions. Then comes a **Dense layer** with 128 units and ReLU activation to learn non-linear feature combinations, followed by a **Dropout layer** with a 0.4 rate to reduce overfitting.

2. The **Output layer** has 3 units (for Science, Mathematics, and History) with a softmax activation to output class probabilities.

3. The model is compiled with **Categorical Cross-Entropy** loss (suitable for multi-class classification), **AdamW optimizer** (learning rate: `2.5e-4`, weight decay: `0.003`, beta_1: `0.9`, beta_2: `0.999`) for better generalization, and **Accuracy** as the evaluation metric.

4. The model was trained for **15 epochs**. It leverages bidirectional context understanding, dropout-based regularization, and AdamW optimization to balance generalization and efficiency in a compact architecture.

Layer (type)	**Output Shape**	**Param #**
embedding_2 (Embedding)	(None, 400, 256)	768,000
bidirectional_2	(None, 256)	394,240
dense_4 (Dense)	(None, 128)	32,896
dropout_2 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 3)	387

Total params: 1,195,523 (4.56 MB)
Trainable params: 1,195,523 (4.56 MB)
Non-trainable params: 0 (0.00 B)

2)Generation Model Architecture

The model follows a sequential architecture:

1. The model begins with an Embedding layer that maps each token index to a 256-dimensional dense vector, with an input shape of `(max_sequence_length - 1,)`. This is followed by an LSTM layer with 192 units and `return_sequences=True`, allowing it to pass a sequence of hidden states to the next layer.

2. Next is a GRU layer with 192 units and `return_sequences=False`, which processes the output of the LSTM and outputs the final hidden state. A Dropout layer with a rate of 0.4 is applied for regularization, followed by a LayerNormalization layer to stabilize and accelerate training.

3. A Dense layer with 128 units and GELU activation introduces non-linearity. Finally, a Dense output layer with `vocab_size` units and a `softmax` activation outputs a probability distribution over the vocabulary.

4. The model uses the AdamW optimizer with a learning rate of 5e-4 and gradient clipping (`clipnorm=1.0`). Mixed precision training is enabled using `LossScaleOptimizer` for numerical stability. The model is compiled with sparse categorical cross-entropy loss and is evaluated using both accuracy and perplexity metrics.

Layer (type)	**Output Shape**	**Param #**
embedding_5 (Embedding)	(None, 255, 256)	3,840,000
lstm_10 (LSTM)	(None, 255, 192)	344,832
gru (GRU)	(None, 192)	222,336
dropout_5 (Dropout)	(None, 192)	0
layer_normalization_5	(None, 192)	384
dense_10 (Dense)	(None, 128)	24,704
dense_11 (Dense)	(None, 15000)	1,935,000

Total params: 6,367,256 (24.29 MB)
Trainable params: 6,367,256 (24.29 MB)
Non-trainable params: 0 (0.00 B)

Text Preprocessing Steps

1. **Remove special characters:** Only keep alphanumeric characters, whitespace, and punctuation (.,!?).
2. **Remove standalone numbers:** Eliminate isolated digits to reduce noise.
3. **Normalize scientific terms:** Replace "co2" with "carbon dioxide" for consistency.
4. **Convert to lowercase:** Standardize the text by making all characters lowercase.
5. **Remove extra spaces:** Trim redundant whitespaces for clean tokenization.

Tokenizer Usage

For Text Generation

1. A single large corpus (`science_corpus`) is created by joining all training texts.
2. `Tokenizer(num_words=15000)` is used to limit the vocabulary to the top 15,000 most frequent words.
3. The tokenizer is fit on the entire joined corpus using `fit_on_texts([science_corpus])`.
4. The final vocabulary size `total_words` is calculated as the lesser of (vocab size + 1) and 15,000.
5. This setup ensures consistent token IDs for word prediction tasks.

For Text Classification

1. `Tokenizer(num_words=max_words)` is initialized with a predefined vocabulary limit (e.g., `max_words = 15000`).
2. The tokenizer is fit on the list of individual training samples `X_train` using `fit_on_texts(X_train)`.
3. This approach captures token frequencies across all input samples for classification tasks, where label supervision is involved.

Model Performance Commentary

Text Classification

1. **Per-Class Performance:**
    a. **Science:** Precision 0.84, Recall 0.90, F1-score 0.87 – slightly lower precision suggests occasional confusion with other subjects.
    b. **Math:** Precision 0.94, Recall 0.93, F1-score 0.94 – most consistently predicted class.
    c. **History:** Precision 0.92, Recall 0.86, F1-score 0.89 – minor recall dip hints at some missed history samples.
2. **Macro and Weighted Averages:** All hover around **0.90**, indicating balanced performance across classes.

Overall: Achieved a strong **90%** accuracy across all three classes, which is excellent for a sequential model, especially considering the interrelation between Science and Math, where certain topics overlap.

Text Generation Performance

1. **Training Accuracy:** Achieved around **41.4%** with a **loss of 3.01**, indicating moderate learning in token prediction.
2. **Perplexity:** The training perplexity is **20.4**, showing that the model is learning but still somewhat unsure about predicting the next word in many cases.
3. **Validation Metrics:**
    a. **Validation Accuracy:** Drops to **33.8%**, which suggests potential overfitting or generalization issues.
    b. **Validation Perplexity: 89.0**, indicating significant struggles when predicting unseen sequences.

Overall: The sequential LSTM model shows potential but falls short in capturing long-range dependencies, leading to higher perplexity. **Transformer models**, with their superior ability to handle such dependencies, could likely achieve a **perplexity under 50** and improve both accuracy and generalization.