**SimpleRNN Model**

**Architecture:**  
The SimpleRNN model consists of three layers. The first layer is an **Embedding** layer with a vocabulary size of **max\_words** and an output dimension of 64. This layer converts the input sequences into dense vectors of fixed size. The second layer is a **SimpleRNN** layer with 32 units, which processes the sequence data. The final layer is a **Dense** layer with 3 units and a **softmax** activation function, which outputs the probabilities for each class.

**Compilation:**  
The model is compiled using the Adam optimizer, which is efficient and widely used for training deep learning models. The loss function is **sparse\_categorical\_crossentropy**, suitable for multi-class classification problems. The model's performance is evaluated using the **accuracy** metric, which measures the fraction of correctly predicted instances.

**Hyperparameters:**

* **max\_words**: Varies based on experiment (500 or 1000) (maximum number of words in the vocabulary)
* **max\_len**: 100 (maximum length of input sequences)
* **epochs**: Varies based on experiment (5 or 10)

The **Embedding** layer is chosen to convert words into dense vectors, facilitating learning. The **SimpleRNN** layer is chosen for its simplicity and effectiveness in handling sequence data. The **Dense** layer with **softmax** activation is used for classification purposes. Adam optimizer is chosen for its adaptive learning rate capabilities, which generally result in better performance.

**LSTM Model**

**Architecture:**  
The LSTM model also starts with an **Embedding** layer with the same configuration as the SimpleRNN model. The second layer is an **LSTM** layer with 32 units, which is more complex than SimpleRNN and capable of capturing long-term dependencies in the sequence data. The final layer is a **Dense** layer with 3 units and a **softmax** activation function, outputting the class probabilities.

**Compilation:**  
Similar to the SimpleRNN model, the LSTM model uses the Adam optimizer, **sparse\_categorical\_crossentropy** loss function, and **accuracy** metric for evaluation.

**Hyperparameters:**

* **max\_words**: Varies based on experiment (500 or 1000)
* **max\_len**: 100
* **epochs**: Varies based on experiment (5 or 10)

The **LSTM** layer is chosen over SimpleRNN for its ability to handle longer sequences and remember information over time, which can lead to better performance in tasks involving sequence data. The same rationale applies for the **Embedding** and **Dense** layers, and the choice of Adam optimizer.

**Experimental Setup**

The models were trained and evaluated using different parameter combinations, including varying splitting ratios, sequence padding lengths, and epochs. The specific parameters explored were:

* **splitting\_ratio**: [0.6, 0.7, 0.9]
* **sequence\_padding\_length**: [500, 1000]
* **epochs**: [5, 10]

**Experimental Results**

The models were trained and evaluated using different parameter combinations, including varying splitting ratios, sequence padding lengths, and epochs. The best performing models for each architecture are identified based on the accuracy achieved on the test set.

A close-up of a table

Description automatically generated

From the experimental results, the best SimpleRNN model achieves an accuracy of 81.05% with a splitting ratio of 0.6, sequence padding length of 1000, and 5 epochs. The best LSTM model achieves an accuracy of 83.23% with the same splitting ratio, sequence padding length, and epochs. Thus, the LSTM model with a splitting ratio of 0.6, sequence padding length of 1000, and 5 epochs is the overall best model based on the achieved accuracy.