Large Language Models (LLMs) are becoming an integral part of our daily interactions, but aligning their responses with human preferences remains a crucial challenge. In a recent project, I worked on predicting user choices based on real-world LLMs chatbot conversations from the Chatbot Arena.

💡 The Challenge:

Users engage in conversations with two anonymous LLMs and select their preferred response. The goal is to build a model that accurately predicts which response a user will favor—essentially bridging the gap between LLM capabilities and human preference.

将Keras的后端设置为jax，这意味着Keras将使用JAX作为其底层计算引擎。JAX是一个由Google开发的高性能数值计算库，特别适合在GPU和TPU上进行加速。

定义一个名为CFG的类，该类用于存储和管理配置参数。

设置预训练模型的名称。（DeBERTa是一种基于Transformer的自然语言处理模型）。

启用混合精度计算，通过结合float16和float32，在保持模型精度的同时提高计算效率和减少内存占用。

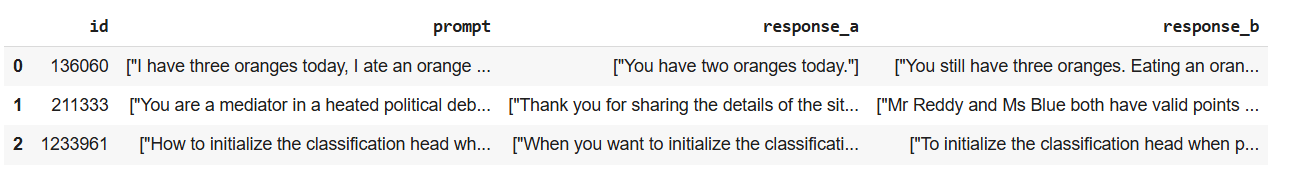
训练数据：



将prompt、response\_a、response\_b列中的字符串形式的列表解析为Python对象。

提取每个列表的第一个元素，并将其存储回原列。

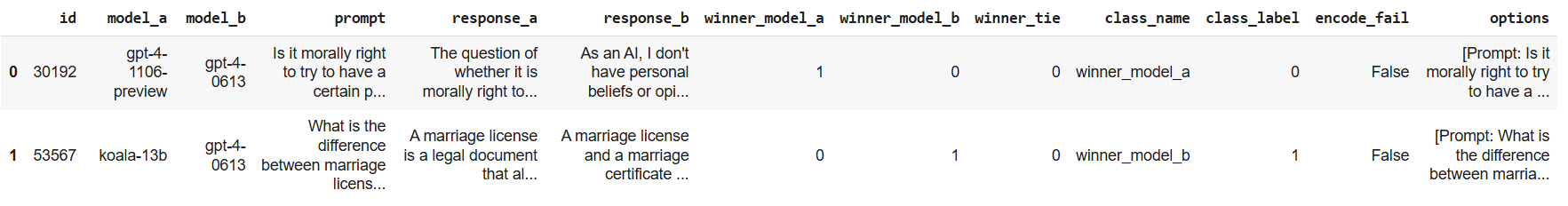
对于response\_a和response\_b列，额外处理了"null"值，将其替换为空字符串。

测试数据：  


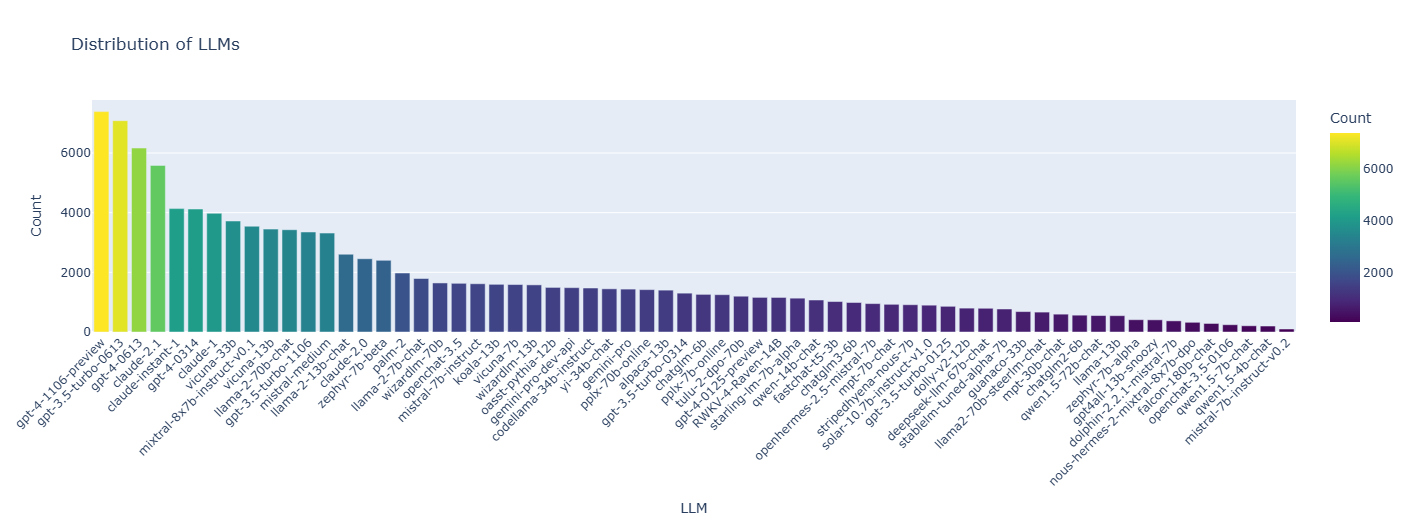
对prompt、response\_a、response\_b进行UTF-8编码和解码，确保它们是有效的字符串。

如果编码或解码失败，将对应字段设置为空字符串，并标记encode\_fail为True。

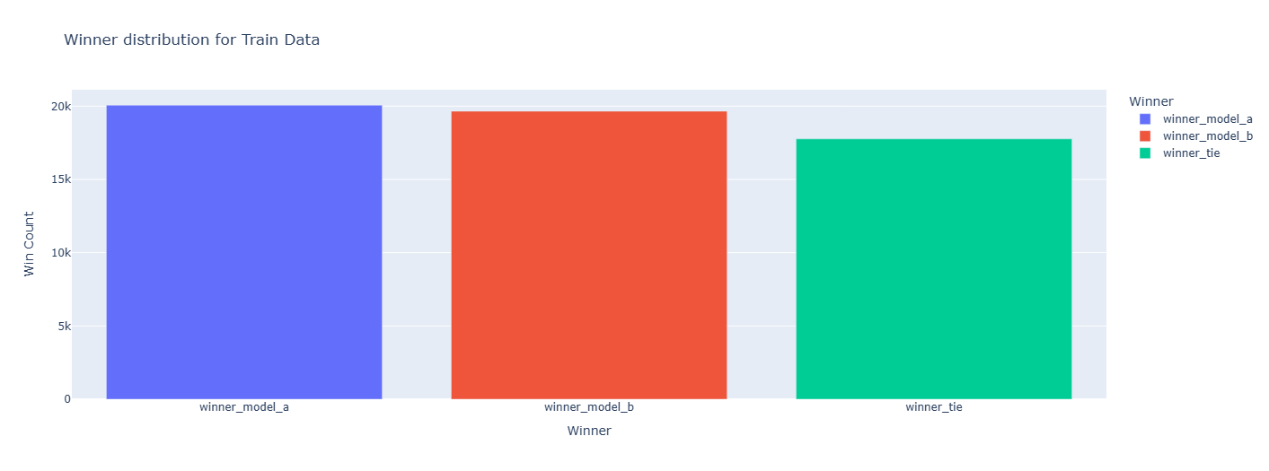
将prompt与response\_a和response\_b组合成两个选项字符串，存储在options字段中。

组合后的dataframe：  


LLM的数据分布：



Winner distribution for Train Data：



可以看到分布相对平均。

将数据集划分为训练集和验证集，训练集占80%，验证集占20%。

使用分层抽样，确保训练集和验证集中class\_label的类别分布与原始数据集一致。

根据预定义的配置（CFG.preset）创建一个DeBERTa V3模型的预处理器。

设置输入序列的长度为CFG.sequence\_length。

预处理器用于将原始文本数据转换为模型输入格式，包括分词、添加特殊标记、填充或截断等操作。

分词（tokenization）：将文本拆分为模型可以理解的子词或标记。

添加特殊标记：如[CLS]（用于分类任务）和[SEP]（用于分隔句子）。

填充或截断：将输入序列调整为固定长度（sequence\_length）。

生成输入ID、注意力掩码（attention mask）等。

构建一个高效的数据加载管道，包括以下步骤：

将标签转换为one-hot编码（如果提供标签）。

将数据转换为tf.data.Dataset对象。

缓存数据（如果cache=True）。

对数据进行预处理（使用preprocess\_fn函数）。

打乱数据（如果shuffle=True）。

将数据分批处理。

提前加载数据（prefetch）。

在模型训练过程中，每次验证集的log\_loss达到新低时，保存模型的权重到best\_model.weights.h5文件中。

如果log\_loss没有改善，则不保存权重。

创建一个Keras指标log\_loss，用于计算多分类任务的对数损失。

适用于标签为one-hot编码格式的情况。

可以在模型训练或评估过程中使用，用于监控模型的性能。

模型结构

**DeBERTa V3骨干网络**：使用keras\_nlp.models.DebertaV3Backbone.from\_preset加载预训练的DeBERTa V3模型。

该骨干网络负责将输入的token序列转换为语义嵌入表示（embeddings）。

**共享权重**：对response\_a和response\_b使用相同的DeBERTa V3模型（共享权重）计算嵌入表示。

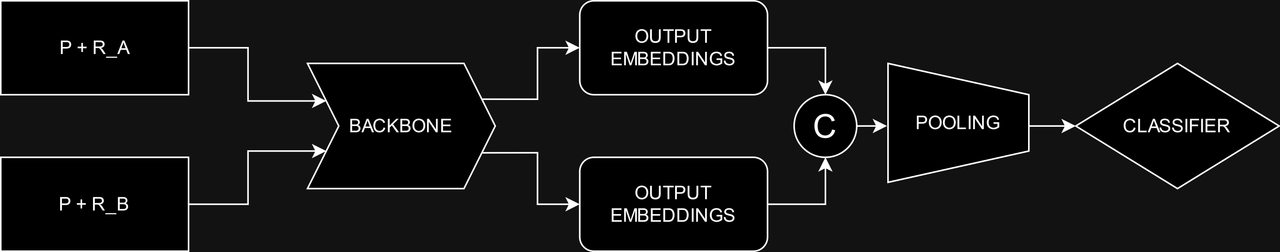
**嵌入拼接**：将response\_a和response\_b的嵌入表示沿最后一维拼接。

**全局平均池化**：对拼接后的嵌入进行全局平均池化，生成固定长度的向量。

**分类器**：使用一个全连接层（Dense）输出3维结果，分别表示：

* winner\_model\_a（response\_a更佳）
* winner\_model\_b（response\_b更佳）
* draw（两个响应相等）

使用softmax激活函数输出概率分布。



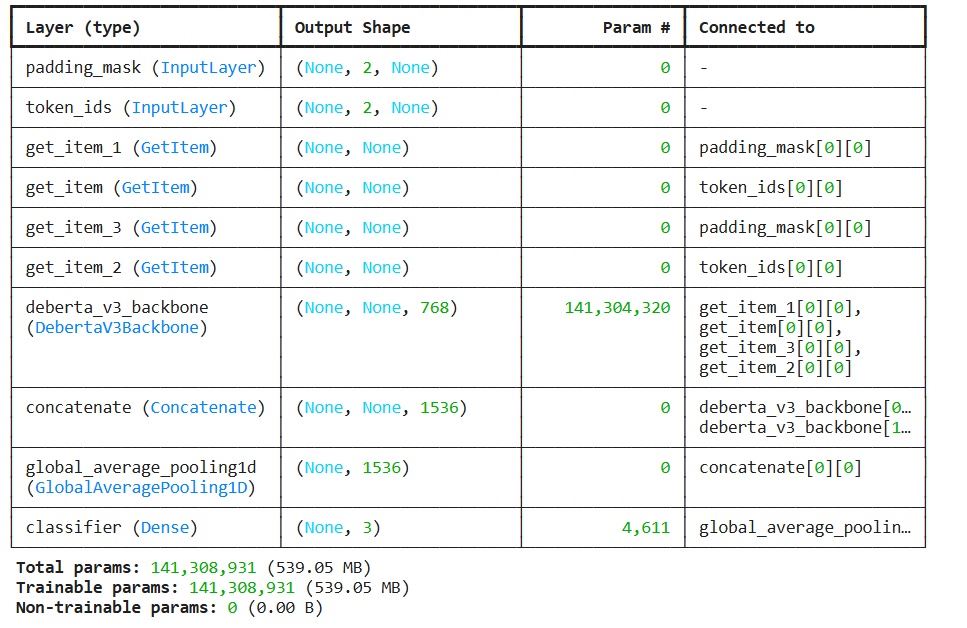
模型编译

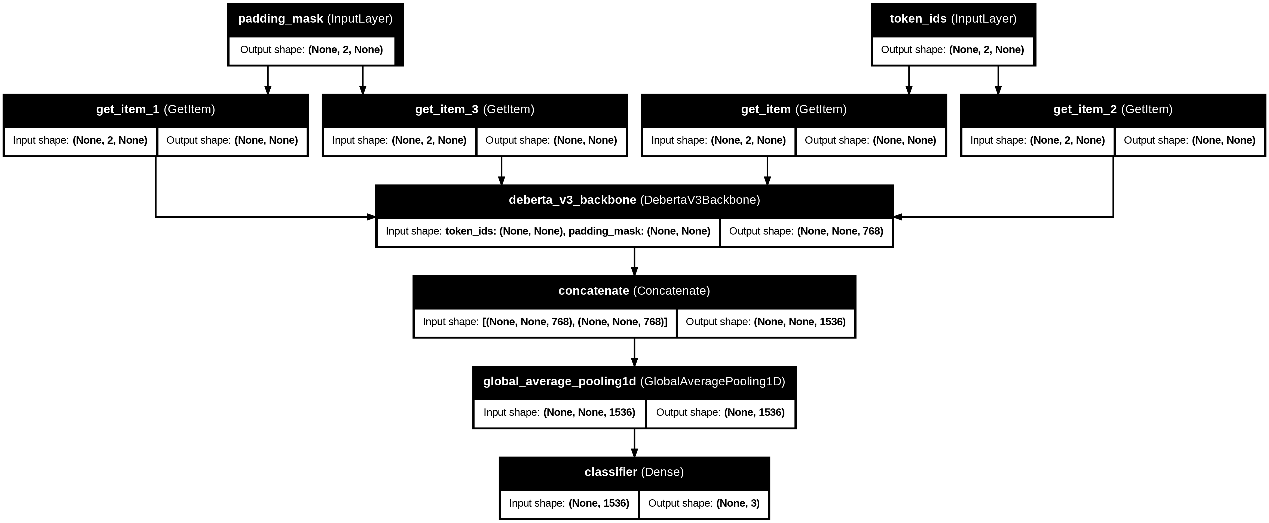
优化器：使用Adam优化器，学习率为5e-6（较小的学习率适合预训练模型的微调）。

损失函数：使用CategoricalCrossentropy，并设置label\_smoothing=0.02以减少过拟合。

评估指标：log\_loss：对数损失。CategoricalAccuracy：分类准确率。

模型参数总结：



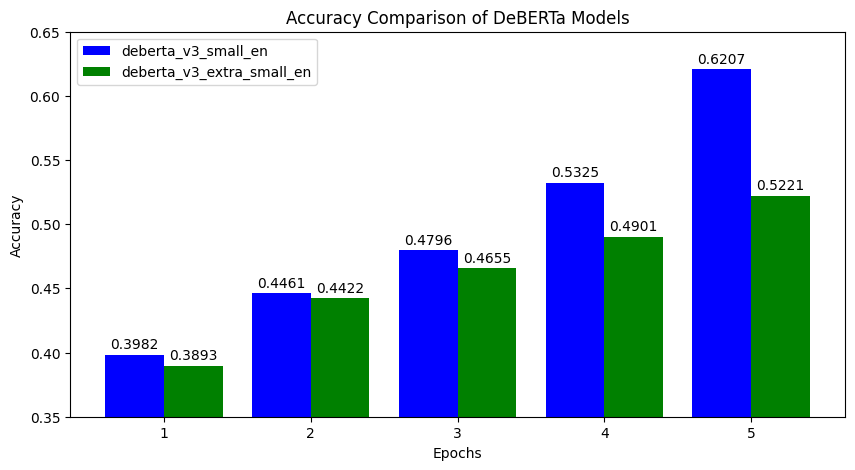


使用训练数据集（train\_ds）训练模型，训练轮数为CFG.epochs。

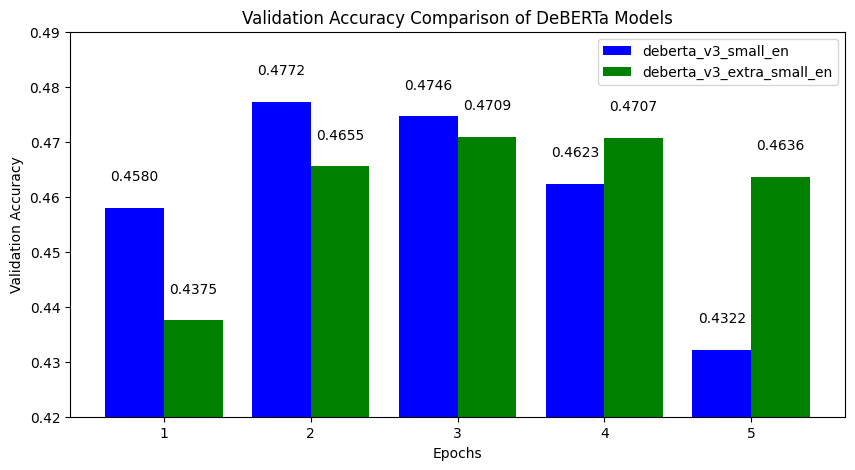
在每个epoch结束后，使用验证数据集（valid\_ds）评估模型性能。

使用回调函数lr\_cb动态调整学习率，并使用ckpt\_cb保存最佳模型权重。

返回history对象，包含训练和验证过程中的损失和指标记录。

训练结果：  


deberta\_v3\_small\_en 在 accuracy 上整体优于 deberta\_v3\_extra\_small\_en，尤其是在后期 epochs 的提升更明显。



deberta\_v3\_small\_en 的 validation accuracy 在初期较高，但在后期略有下降，而 deberta\_v3\_extra\_small\_en 的 validation accuracy 更加稳定。