文本载体毒性预测模块

对样本进行预测在我们的系统中分为验证性预测与测试性预测，前者为在模型训练阶段通过对验证集进行预测以评估模型此时的学习效果，后者为在模型测试阶段通过对测试集进行预测以评估模型的泛化能力及训练后的预测能力。我们引入两个指标以量化并且评估预测阶段的模型效果，分别是准确度（Accuracy）与 F1 得分（F1 Score），计算方式如下所示：

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \tag{1}$$

$$Precision = \frac{TP}{TP+FP}, \tag{2}$$

$$Recall = \frac{TP}{TP+FN}, \tag{3}$$

$$F1 = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}} = \frac{2*Precision*Recall}{Precision+Recall}, \tag{4}$$

在上述公式中，其中涉及到的变量含义如下：

- **TP**：True Positive。预测为 1，实际为 1，预测正确的样本数量。
- **FP**：False Positive。预测为 1，实际为 0，预测错误的样本数量。
- **FN**：False Negative。预测为 0，实际为 1，预测错误的样本数量。
- **TN**：True Negative。预测为 0，实际为 0，预测正确的样本数量。

在这一模块中，训练集与验证集的划分尤其重要，之后我们在测试数据集完成对模型的性能测试，框架如下所述：

- **模块整体框架**

```
if valid_split > 0:
    train_size = int((1. - valid_split)*len(X["input_ids"]))
    X_train, y_train = {"input_ids": X["input_ids"][:train_size], "token_type_ids": X["token_type_ids"][:train_size], "attention_mask": X["attention_mask"][:train_size]}, y[:train_size]

    X_dev, y_dev = {"input_ids": X["input_ids"][train_size:], "token_type_ids": X["token_type_ids"][train_size:], "attention_mask": X["attention_mask"][train_size:]}, y[train_size:]
else:
    X_train, y_train = X, y
    X_dev, y_dev = X_test, y_test
```

```
9.
10.     predictions = (model.predict(X_test, verbose=0)).round()
11.     y_pred = np.argmax(predictions, axis=1)
12.     C = confusion_matrix(y_test, y_pred)
13.     logger.info("confusion_matrix: {}".format(C))
14.     acc = accuracy_score(y_test, y_pred)
15.     bert_acc.append(acc)
16.     logger.info("Test accuracy: {}".format(acc))
17.     logger.info("Test macro_f1: {}".format(
18.         f1_score(y_test, y_pred, average="macro")))
19.     pre = precision_score(y_test, y_pred, pos_label=1)
20.     bert_pre.append(pre)
21.     logger.info("Test precision: {}".format(pre))
22.     recall = recall_score(y_test, y_pred, pos_label=1)
23.     bert_recall.append(recall)
24.     logger.info("Test recall: {}".format(recall))
25.     f1 = f1_score(y_test, y_pred, pos_label=1)
26.     bert_f1.append(f1)
27.     logger.info("Test f1 score: {}".format(f1))
```

1. 检查是否需要验证集

```
1. if valid_split > 0:
```

检查 valid_split 是否大于 0。如果 valid_split > 0，则表示需要划分验证集。

2. 划分训练集和验证集

```
1. train_size = int((1. - valid_split) * len(X["input_ids"]))
2. X_train, y_train = {
3.     "input_ids": X["input_ids"][:train_size],
4.     "token_type_ids": X["token_type_ids"][:train_size],
5.     "attention_mask": X["attention_mask"][:train_size]
6. }, y[:train_size]
7.
8. X_dev, y_dev = {
9.     "input_ids": X["input_ids"][train_size:],
10.     "token_type_ids": X["token_type_ids"][train_size:],
11.     "attention_mask": X["attention_mask"][train_size:]
12. }, y[train_size:]
```

- 计算训练集的大小 train_size 为总数据大小的 (1 - valid_split)。
- 使用切片操作将数据集 X 分为训练集 X_train 和验证集 X_dev，并分别提取对应的标签 y_train 和 y_dev。其中 X 是包含输入 ID、token 类型 ID 和注意力掩码的字典，y 是标签。

3. 使用测试集进行评估

```
else:
    X_train, y_train = X, y
    X_dev, y_dev = X_test, y_test
```

- 如果不需要验证集 (valid_split <= 0)，就直接使用所有数据 X 和 y 作为训练集。
- 使用测试集 X_test 和 y_test 作为验证集进行后续评估。

4. 模型预测并评估

```
acc = accuracy_score(y_test, y_pred)
bert_acc.append(acc)
logger.info("Test accuracy: {}".format(acc))
```

计算并记录测试集的准确率 accuracy_score 并添加到 bert_acc 列表。

```
logger.info("Test macro_f1: {}".format(f1_score(y_test, y_pred, average="macro")))
```

计算并记录 macro 平均的 F1 分数。

```
pre = precision_score(y_test, y_pred, pos_label=1)
bert_pre.append(pre)
logger.info("Test precision: {}".format(pre))
```

计算并记录 pos_label=1 类别的精确度 precision_score 并添加到 bert_pre 列表。

```
recall = recall_score(y_test, y_pred, pos_label=1)
bert_recall.append(recall)
logger.info("Test recall: {}".format(recall))
```

计算并记录 pos_label=1 类别的召回率 recall_score 并添加到 bert_recall 列表。

```
f1 = f1_score(y_test, y_pred, pos_label=1)
bert_f1.append(f1)
logger.info("Test f1 score: {}".format(f1))
```

计算并记录 pos_label=1 类别的 F1 分数 f1_score 并添加到 bert_f1 列表。

通过上述这些操作，我们全面评估了模型在毒性文本预测任务中的性能，有助于理解模型在不同评价指标（准确率与 F1 得分）下的优劣，从而进一步改进和优化模型参数及架构。