工程实践与科技创新 IV-J 课程作业最终报告

张晨阳　肖真然

目录

1 作业简介

在本次课程作业中，我们探讨了简单大语言模型在情感分类任务上的能力。通过选取、制作数据集，微调 GPT2、BERT 等预训练模型，在多个现行的文本情感分类任务数据集上取得了一定的效果。在第一部分，本团队针对情感分类任务在 GPT2 框架上进行了探讨；在第二部分，本团队进行了模型工作、训练评估方案改进、数据集拓展等一系列工作，从更深、更广地角度理解简单大语言模型在情感分类任务上的表现。

2 实验环境

基础设施 Linux Ubuntu, CUDA used 4090, Windows 10

深度学习框架 Pytorch, Transformers

预训练模型 `GPT2LMHeadModel.from_pretrained("gpt2")`[1]
`BertForMaskedLM.from_pretrained("bert-base-uncased")`[2]
`BertForMaskedLM.from_pretrained("bert-base-chinese")`[3]

3 第一部分

3.1 任务分析

3.1.1 问题预调研

在文本情感分析这一研究领域中，文本情感分类是一项最为基础的工作。一般以一段文本作为输入，输出为形如 joy、sadness 的情感类别描述词。在一些具体的任务上，数据集往往将文本情感划分为若干个大类别，并使用数字序号与之建立映射，作为数据集标签。

本次作业的第一部分采用的模型是 117M 参数量版本的生成式模型 GPT2。这一模型版本相对较为早期，参数量有限，不能期望模型能够做到分析极细粒度的丰富语义，故在情感主体提取、针对具体属性的细粒度情感分析等任务上很难获得很好的效果。因此，我们仅针对文本总体情感分类这一任务对模型进行微调训练。

尽管 GPT2 模型本身是一个生成模型，但借助提示词工程等一系列方法，可以让模型拥有执行分类任务的能力，从而完成我们期望的文本情感分类任务。

3.1.2 中期分析

根据前期推进的反馈，我们发现了一些问题。为了维护第一部分工作的完整性，将在后续章节中统一介绍全部的数据集与模型训练工作，本小节只简单说明结论。

本小节将对以下问题展开描述：

- 前期使用的三类数据集存在一定的内外部缺陷，导致最终效果不佳；
- 由于 GPT2 本身能力缺陷，分类问题在不经过微调的预训练模型上几乎无法得到任何有效结果，微调效果难以察觉。

数据集缺陷 在前期测试中采用了三个情感多分类的数据集，其具体信息可以参考 3.2.1 板块。事实上，经过微调的 GPT2 模型在这些数据集上的表现并不及预期，包括部分数据之间的极度不平衡对选择题作答的影响、不同数据集间相同情感标签的迁移效果差等情况。

生成能力待开发 在第一部分中我们只使用 117M-GPT2 作为预训练模型框架，在前文中已经说明该模型相比于现行主流大模型部分性能的缺陷情况。而前期工作没有直接使用简单的续写生成任务，而是用提示词工作实现分类任务，这对模型的理解能力而非生成能力提出了更大的要求，因而进一步放大了 GPT2 框架的部分缺点。

基于以上问题，我们在第一部分前期工作的基础上进一步做了以下工作：

- 引入个性化数据集制作代表性情感文本；
- 发挥模型生成式能力辅助数据集迁移。

3.2 数据集调研与选取

3.2.1 分类用数据集

文本情感分类在 NLP（自然语言处理）中是一个被广泛考察的任务课题，现行有许多开源的相关数据集可供使用。在广泛地检索了各种相关数据集后，我们基于课程作业任务将其分为以下几类比较有代表性的数据集类型。

倾向分析 这一类数据集是占比最多的，数据的文本来源主要是各类网站的评价文本，如商品评价、影片评价等。这类数据的标签通常为二类或者三类，分别是正面情感（Positive）、负面情感（Negative），以及可能的中性情感/陈述。这种分类方法在市场调研等实际应用中会有很高的效益，但过于简单的分类并不适合本次作业中的生成模型，且评判的力度有限。在本次作业中只作为可供使用的文本集合。

细粒度情感分类 这一类数据相对较少，数据的文本来源主要是各类相对简单的日常表达文本，加上部分评论类文本，质量整体相对参差。这类数据集的标签通常为多类，比较常见的有 6 类、13 类等，一般对应着 joy、sadness、love 等大类情感描述词。本次作业将主要使用这类数据集。经过有效筛选，我们从 Kaggle 上选取了三个数据集：

1. `Emotions in text`[4] 6 类情感分类数据共 21459 条；
2. `Emotion Detection from Text`[5] 13 类情感分类数据共 40000 条；

3. `emotion analysis based on text`[6] 13 类情感分类数据共 839555 条，其中 neutral 类占 80.3%，其余数据共 165017 条。

超细粒度情感分类 在文本情感分析领域，需要特别了解一个数据集——Google 发布的大规模数据集 GoEmotions[7]。这一数据集是上述细粒度情感分类的特例，文本来源于 Reddit 评论，而标签高达 28 类，远超其他数据集。尽管该数据集本身能够导向高精度地情感辨析分类，但考虑到 GPT2 适应度、与其他数据集的兼容等多方面的问题，仅作为可供使用的文本集合，以及情感划分的参考性指标。

3.2.2 生成用数据集

在前期工作的基础上，我们进一步检索了更加广泛的文本类型数据集，并决定引入一部分具有显著情感特征的特殊文本数据集来强化模型的情感分析能力。根据不同的需要，此类数据集可能会有不同的使用方法，包括但不限于直接截取单句/段落作为特定标签的数据；投入文本生成式模型批量生产句法表意模糊、情感特征强烈的特殊数据集。根据理论分析，经过微调生成模型获得的特殊数据集可以更加适应文本理解能力相对有限的旧版本框架，以数据集的形式调整微调的注意力到情感分类问题上，进而在广域数据范围内获得相对好的效果。

现有数据集模糊化 为了更好地获得充满情感表意的语意模糊数据集，生成用数据集可以直接复用原有数据集的部分数据。在排除掉本身损坏的数据、去除过于不平衡的数据比例后，可以对每一个标签单独训练一个生成式模型，用来给分类模型提供数据集扩充。

我们在理论上对这种做法的优缺点进行了分析：

- 作为一种数据增广，能够扩张模型的适用范围；
- 数据集指向模糊，在多个数据集综合上会有更好的迁移效果；
- 相比于传统的数据增广，更能发挥生成式模型的优势；
- 如果模型生成能力不足，会反向干扰分类模型。

特殊小型数据集 在这一部分，我们引入了 Love Letters[8]、Stress Detection from Social Media Articles[9] 等一系列的小型数据集。这些数据集的特征在前文已提及，是具有显著情感特征/倾向的特殊文本数据集。第一个数据集是一个超小型的情书文本数据集，值得一提的是，这是我们最初调研的一个数据集；另一个数据集是一个相对有些规模的关于压力情绪的文本数据集。这类数据集可以引入用来进一步提升模型的延展性，但不可避免的会影响模型在原有训练集与验证集上的性能。因而在后续评估中，这方面的延展性评估会由非纯分类的方式进行评测。

进阶数据集 在这一部分，我们引入了包括 Short Jokes Dataset[10] 在内的一些小型数据集。这些数据集是在以上数据集能够充分实现预期目标后的追加数据集，对模型的文本理解能力有更高的要求，在预测上基本超过了 GPT2 所支持的语义功能。例如所举例的这一数据集，内容为一系列简短的笑话。在人脑理解中，这些语段是显而易见的“有趣的笑话”，能够带来显而易见的 funny 类似的情绪，但对于情感分析模型而言，这种文本多了一层理解转译，就显得困难重重。因此对于这类数据我们仅做了简单的数据加工、并人工进行了简单的评析。

3.3 模型训练

3.3.1 训练数据预处理与制作

为了使得 GPT2 生成模型能够适应分类问题的需求，本次作业使用的数据集需要经过提示词的预处理加工。主要分为以下两个方面：适应分类问答的提示词包装工程，以及多个数据集在评价标准的统合工程。

提示词：一问一答 分类问题可以被显式地表示为一套选项固定的选择题。对于生成模型而言，为了方便后续测试的可扩展性，本次作业暂时并不通过截取特征后加装分类网络结构，而是简单的通过提示词文本设计 question+answer 的文本数据组合。由此，训练所用的样本格式如下：

```
question = "Please read the given text, analyze its sentiment, and select one of the following emotion options: empty, sadness, excitement, neutral, fear, surprise, love, amusement, annoyance, joy, boredom, relief, anger. \nThe text is: "
answer = "\nThe answer is:"

full_text = question + texts[i] + answer + ' ' + emotions[i]
```

测试所用提示词如下：

```
prompt = question + texts[i] + answer
```

模型仅预测一个单词的 id，用 tokenizer 还原提示词 + 预测结果的完整文本：

```
with torch.no_grad():
    outputs = model(prompt_tensor)
    predictions = outputs[0]
predict_id = torch.argmax(predictions[0,-1, :]).item()
text = tokenizer.decode(prompt_encoded + [predict_id], skip_special_tokens=True)
```

可以根据经验与测试显见，这类提示词在 Chat GPT 等大模型上会有很有效的表现，基本可以做到对简单语句情感分析的正确回答。由此推理在 GPT2 上应该有相对应地弱化表现。两代模型之间的具体的进一步比较不是本文所探讨的关键，不进一步论述。

单分类数据集统合 由于采用相同格式的输入文本，对于数据集需要进行统一的格式化，一方面是原数据读取格式读取的统合，另一方面是原标签标记描述词的统合。

经过比较数据集发现，本次采用的三个单分类数据集均采用 csv 格式，通过简单地选列与列名统一，可使其结构达到一致。另一方面，两个 13 类数据集采用完全一致的标签构成，且完全包含了 6 类数据集的标签，仅需要对同义词的不同表达——例如 happiness→ joy——进行统一，这里以 GoEmotions 数据集标签为标准，统一了三个数据集的标签。

具体而言，6 类标签分别是 anger、fear、joy、love、sadness、surprise，13 类标签则是在 6 类的基础上增加了 amusement、annoyance、boredom、empty、excitement、neutral、relief。这一点在上一小节的 question 文本中已经有所体现。

由于单分类数据集本身规模较大且 13 类的模型存在部分类别的显著不平衡，在使用中本次作业采取了截取小部分数据集（各 800 条）（总数据条数约为 10k 级）的方式训练模型。

生成用数据集制作 本次作业选取的生成用数据集为在单分类的基础上制作，分别制作了三数据集上合并并打乱的六分类数据集，以及二数据集上合并并打乱的十三分类数据集。数据集整体平衡，适合作为训练测试集。接下来以三数据集上合并打乱制作六分类数据集为例说明制作过程：

1. 从三个原数据集上提取 6 类标签、各自规模为 800 的子集（沿用单分类数据集）。这一步骤是为了保持总数据规模在 k 10k 的适应 gpt2 模型框架的规模尺度，并极大地削减数据集的不平衡程度；
2. 使用 python 库合并并打乱三个数据集，从而得到一个混合数据集。这一步骤是为了让后续训练过程更合理，保持在数据来源上的相对平衡；
3. 对于该数据集，以一定比例划分出训练集与测试集，本例中采取 8:2。最终训练集的大小为 10968；测试集的大小为 2742。

3.3.2 单分类训练

针对单分类数据集，我们设计了训练与测试代码。训练使用了 Transformers 提供的 Trainer 类来简单实现训练过程，在后续项目中，不排除使用更加基础性的工作来支持更加复杂的深度学习设计方法。

采用部分参数如下：

```
1 training_args = TrainingArguments(
```

```
    output_dir='./model/gpt2_trained',
    num_train_epochs=3,
    per_device_train_batch_size=1,
    save_steps=1000,
    learning_rate=1e-4,
    overwrite_output_dir=True,
)
```

完整代码见附件。

3.3.3 生成-分类双向训练

针对生成用数据集，我们同样设计了类似的训练与测试代码，代码的具体实现方式基本延续了单分类的框架。需要说明的是，其中分类部分的代码与单分类训练重用一套代码组件，生成部分的代码也只需要在单分类训练代码上稍加修改，修改内容主要为去除多余的问答 prompt，即在代码中直接使用 texts 作为 full_text。

一般而言，训练分为两个阶段：

1. 先通过现有的同调前数据混合集来训练生成模型；
2. 再通过生成模型生成可用的简单分类数据集引入分类模型进行训练，此时学习率应该设置到较低水平以免干扰原数据集上的效果。

在各方面有余裕的前提下，可以进一步考虑将模糊语句引入生成模型反复以上操作，兼顾模型性能和延展度来微调模型。

在设计训练生成模型的过程中，实验发现投入小规模（k 10k 级别）数据集并不能获得优质的数据生成模型，在稍长句段的生成中，缺乏上下文联系的句段往往容易出现情感的消散、缺失甚至是无逻辑的转变，这类文本不但无法提供较好的训练效果，反而容易污染原本的模型。因而在数据生成中，往往仅使用 10 15token 的长度的文本续写生成来作为生成数据，如此生成的数据往往在在靠前的位置就爆发出显著的情感特征词，相对易于判明，而在数据集调研中，可以认为这种文本数据占据优质数据的主流。

此外还有生成文本重复等一系列的小问题，我们在数据处理中设计了一系列的小型 python 脚本来解决这部分问题，在 readme 中会有简单描述，报告中略去。

3.4 微调效果展示

在这一小节将展示微调训练前后模型处理情感分类问题上性能的差异。作为初步比较，先使用手动判断准确率（Accuracy）作为唯一指标，在后续涉及更加精度的模型优化时，可能会根据实际需要引入更加复杂的评判机制。

3.4.1 单分类效果

在微调训练前，生成式模型对问题的理解能力可以认为完全为 0，生成的 Anwser 部分几乎全为字母 “i”（可能是因为 “i” 为英文中出现频率最高的字母）。经过一系列初步微调后的结果如下：

Dataset	Size (pre label)	Test	Accuracy
EIT	800	EIT	0.905
EIT	800	EDFT	0.149
EDFT	800	EIT	0.26
EDFT	800	EDFT	0.269
EDFT	800	EABOT	0.115
EABOT	800	EIT	0.068
EABOT	800	EDFT	0.213
EABOT	800	EABOT	0.914
ALL	800*3	EIT	0.895
ALL	800*3	EDFT	0.154
ALL	800*3	EABOT	0.069

从测试结果来看，GPT2 能在 6 分类场景下表现出良好的性能，但在 13 分类的场景下能力显著下降。特别地，分析认为在 EABOT 数据集上自训练验证的高准确率是由于本身数据集的极度不平衡，模糊时选择 neutral 可以解决 80% 的问题，故而缺乏一定的讨论价值，在后续的实验中也会逐步削弱这部分的数据影响。

关于以上结果，我们经过分析，提出了以下可能：

1. 数据集简单聚合的情况下，旧 6 类占据数据集优势，即不平衡而影响数据集；
2. 文本情感的细粒度多分类问题确实对于 GPT2 来说难度偏大，在扩展 13 类时导致断崖式的困难；
3. 数据集之间尽管采用类似的标签模式，但数据集本身之间依然存在许多需要调和的地方，如数据集整体的 “距离”、在一些模糊句段中数据集区分情感的不同等原因。

3.4.2 双向训练效果

生成数据可用性测试 首先在原数据训练模型上测试生成数据集，在生成数据训练模型上测试原数据集。这两个数据将会说明原数据与生成数据在模型识别上的相似度，进一步说明生成数据的可靠程度。

- 在原数据训练模型上测试生成数据集，最终得到的准确率是 **0.59561**；
- 在生成数据训练模型上测试原数据集，最终得到的准确率是 **0.56018**。

生成数据优化测试 接下来，在原数据训练模型上额外引入生成数据进行训练。此处较好的做法是将新生成数据打散洗入训练过程，以免先后训练对模型造成指向性干扰。但为了说明生成数据对训练模型的优化效果，本次直接在经过微调训练的模型上增加 epoch 微调，为了避免干扰，初始学习率被显著降低。

经过一次训练，准确率从 0.75784 来到了 **0.75821**。

4 第二部分

4.1 模型工作

4.1.1 GPT2 与 BERT

在第一部分的任务中，我们已经充分讨论了 GPT2 模型在文本理解能力上的相对不足，尽管已经有开源的工作表明可以通过修改网络输出端等方式来获得不依赖额外文本理解成本的 GPT2 分类模型框架，但依然无法回避在同时代同规模模型中 GPT2 更擅长生成的事实。出于这种考虑，在第二部分中，我们提出引入 BERT 这一更加适应情感分析的预训练模型框架来展开实验任务。具体而言采用的是 Google 发布的 bert-base-uncased 模型。

BERT 和 GPT2 是经常被拿来比较的经典模型框架，其中 BERT 的核心是双向 Transformer 编码框架，相比 GPT2 在生成问题上效果并不凸显，但相对有着更强的文本理解能力，更好的下游任务适应性，在处理如情感分类的具体下游任务上往往略有优势。因而本次作业考虑引入 BERT 来替代 GPT2 进行分类模型框架上的一些探索。

基于以上考虑，使用制作好的混合数据集在 BERT 上进行微调测试。其中微调前 BERT 的准确率可以在 6 类数据集上达到 **0.14∼0.17**，13 类数据集上达到 **0.06** 左右。而微调后的结果如下：

Dataset	Epoch	Accuracy
6-label	1	0.73231218
6-label	3	0.78482859
6-label	5	0.77388767
13-label	1	0.60276890
13-label	3	0.63684771

可以看到在混合集合上，BERT 的表现整体略优于 GPT2，可以高出约 $2 \sim 4$ 个百分点。

4.1.2 token 编码差异

BERT 模型本身使用的 Tokenizer 方法是基于 WordPiece 的分词方法。这是一种子词粒度的分析方法，构建一个子词词典并通过形如 [CLS][SEP] 等 tag 来提示序列边界。这种方法本身是一种非常具有广泛适用性的分词方法，但单就本次作业而言，在情感分类上可以通过一些小 trick 来实现语义划分的优化。

经过数据分析，认为大部分数据的情感明显且集中于个别词汇，所以我们推测将这些个别词汇完整保留不经过子词分解，会更加有利于情感判别。只需要配置一个情感爆发词汇词典，在子词分词器外层多套一层词典就可以实现这一优化。

基于以上分析，可以通过简单的代码段，将人工小词典与 BERT 本身的 WordPiece 分词结合使用，即当出现小词典中的单词时完整记录词段，剩余部分再使用 WordPiece 分词。在本次实验中，我们只设置了简单的小词典，将 6 类/13 类标签本身作为重要词段完整记录。核心代码如下：

```
class CustomWordPieceTokenizer:
    def __init__(self, custom_vocab, pretrained_model_name_or_path='bert'):
        self.custom_vocab = set(custom_vocab)
        self.bert_tokenizer = BertTokenizer.from_pretrained(
    pretrained_model_name_or_path)

    def tokenize(self, text):
        words = text.split()
        tokens = []
        for word in words:
            if word in self.custom_vocab:
                tokens.append(word)
            else:
                tokens.extend(self.bert_tokenizer.tokenize(word))
        return tokens
```

经过以上处理后，依然在混合集合上测试，最终结果如下：

Dataset	Epoch	Accuracy
6-label	1	0.74471189
6-label	3	0.78920496
13-label	1	0.61714590

可以看到在混合集合上，经过极简处理，数据集的准确率可以比同期明显高出 1 个百分点左右，这是非常显著的提升，相应地也印证了之前数据分析的结论。

4.1.3 注意力机制分析

BERT 所使用的注意力机制是 Transformer 框架中的自注意力机制（Self-Attention Mechanism），其具体内容在先前课堂与任务中均有涉及，这里不多叙述。以下是单头自注意力的核心公式，实际中有多头注意力的使用，但原理算式是一致的。

$$Attention = \frac{QK^T}{\sqrt{d_k}} Output = Attentino \cdot V$$

从宏观网络视角，BERT 自注意力机制的效果主要体现于不同 token 间的信息联合捕捉上。根据先前的人工数据分析，本次作业中涉及的绝大多数数据在情感表达上都呈现出极短词组上的快速爆发，在上下文上没有明显的需求。因而我们最终没有引入新的方式改进注意力机制。

4.2 双向训练评估

在 BERT 模型中，我们依然延续了 3.3.3 中关于双向训练的探究。由于第一部分已经说明了这一训练的可能性和可行性，在第二部分中，改为由 BERT 负责分类模型构建，由 GPT2 负责生成模型构建，并在 6 类数据集上展开了调参工作。最终可以在基础 epoch5 代之内精修得到准确率为 **0.87892050** 的模型，具体的调参工作经历了反复尝试，最终这个模型被保存在`./model/berf_6f3_2` 目录下。

4.3 数据集拓展

4.3.1 中文场景

我们选择 Google 发布的 bert-base-chinese 作为预训练模型，分别在二分类和多分类的场景下进行了微调尝试，任务模式同第一部分相似。

二分类：ChnSentiCorp[11] ChnSentiCorp 是一个中文情感分析数据集，包含酒店、笔记本电脑和书籍的网购评论，共 12000 条数据。数据有两种标签，正向和负向，经过清洗整理后用“正”代表正向，“负”代表负向。本次微调使用了原数据集第 1-1000 条作为训练数据，第 1001-1100 作为测试数据。训练所用样本格式如下：

```
question = "请分析给定文本的情感倾向，正向请选择正，负向请选择负。\n文本如下："
answer = "\n答案是："

full_text = question + texts[i] + answer + emotions[i]
```

采用部分训练参数如下：

```
training_args = TrainingArguments(
    output_dir='./model/bert_zh_2_retrained',
```

```
    num_train_epochs=3,
    per_device_train_batch_size=1,
    save_steps=500,
    learning_rate=1e-4,
    overwrite_output_dir=True,
)
```

测试所用 prompt 如下：

```
prompt = question + texts[i] + answer + "[MASK]"
```

由模型来填写 [MASK] 部分，填入项为一个 token：

```
unmasker = pipeline('fill-mask', model='bert_zh_2_retrained', tokenizer=
    tokenizer)
output = unmasker(prompt)
predict_emotion = output[0]['token_str']
```

完整代码见附件。

微调后模型在测试数据上的准确率为：**0.82**。

多分类：OCEMOTION[12] OCEMOTION 是包含 7 个分类的中文细粒度情感性分析数据集，共 35693 条数据，其中 7 个情感类别分别为 happiness、sadness、anger、like、surprise、fear、disgust。经过清洗整理后使用标签如下：乐，哀，怒，爱，惊，惧，恶。本次微调使用了原数据集第 1-1000 条作为训练数据，第 1001-1100 作为测试数据。训练所用样本格式如下：

```
question = "请分析给定文本的情感，并从以下选项中选择一个：乐，哀，怒，爱，惊，惧，
    恶。\n文本如下："
answer = "\n答案是："

full_text = question + texts[i] + answer + emotions[i]
```

训练参数与测试代码与二分类任务大体一致。完整代码见附件。

微调后模型在测试数据上的准确率为：**0.40**。

将 num_train_epochs 改为 5，重新微调原模型，最终准确率为：**0.37**。说明增加训练 epoch 对模型并无提升效果。

从结果来看，BERT 用于中文情感分析任务，在二分类的场景下的表现显著优于七分类，但都逊于 BERT 在英文任务中的表现。分析可能原因如下：

- BERT 中文预训练模型初始质量低于与英文预训练模型；

- BERT 中文分词为字符级分词，这可能会影响到模型对于中文词组的理解；
- 中文数据集质量欠佳。在清洗整理数据集的过程中，发现部分数据初始标注存在争议，但为了让微调结果呈现出最自然的效果，并没有对其进行修改。

由此可见，本项目中在中文情感分析的进展还相对落后，存在较大的提升空间。投射到更大的视角，中文情感分析由于各方面的原因可能会有更大的困难，需要通过一定的优化方法来取得较好的效果。

4.3.2 小样本私域集

小样本私域集的引入可以参见第一部分中 3.2.2 节，其中引入的以 Love Letter 为代表的小规模样本集已经被使用以增强模型的广度与延伸性。

5 总结

在本次作业的第一部分中，我们以简单大语言模型在情感分类课题上的能力为研究课题，以 117M 轻量级 GPT2 模型为基础开展了数据集与模型微调工作，最终在三个情感分类数据集的组合集上取得了一定的成果，但缺陷也很明显。为了得到更好的效果，我们利用特殊小型数据集对模型进行了生成任务的微调，并将微调后的模型生成的文本加入新的混合分类数据集重新微调模型分类，最终得到了明显的提升。在第二部分中，我们换用 BERT 模型，尝试了新的 token 编码方式，也在中文场景下进行了实验，认识到了不同模型、不同编码方式、不同数据集对微调效果的影响。

总的来说，我们通过本次课程作业，尝试学习了对简单大语言模型（GPT2、BERT）的微调训练，也探索了更多细节。收获颇丰。最后感谢邓老师和助教们在本次作业中提供的帮助！

6 References

[1] Open AI. openai-community/gpt2. https://hf-mirror.com/openai-community/gpt2, 2024. Online; last updated in 2024.

[2] Google. google-bert/bert-base-uncased. https://hf-mirror.com/google-bert/bert-base-uncased, 2024. Online; last updated in 2024.

[3] Google. google-bert/bert-base-chinese. https://hf-mirror.com/google-bert/bert-base-chinese, 2024. Online; last updated in 2024.

[4] Ishant. Emotions in text. https://www.kaggle.com/datasets/ishantjuyal/emotions-in-text, 2020. Online; last updated in 2020.

[5] Pashupati Gupta. Emotion detection from text. https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text, 2021. Online; last updated in 2021.

[6] Sima Anjali. emotion analysis based on text. https://www.kaggle.com/datasets/simaanjali/emotion-analysis-based-on-text, 2024. Online; last updated in 2024.

[7] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. GoEmotions: A Dataset of Fine-Grained Emotions. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[8] Sreeram Venkitesh. Love letters. https://www.kaggle.com/datasets/fillerink/love-letters, 2020. Online; last updated in 2020.

[9] mexwell. Stress detection from social media articles. https://www.kaggle.com/datasets/mexwell/stress-detection-from-social-media-articles, 2024. Online; last updated in 2024.

[10] The Devastator. Short jokes dataset. https://www.kaggle.com/datasets/thedevastator/short-jokes-dataset, 2023. Online; last updated in 2023.

[11] 谭松波，中国科学院，千言数据集. Chnsenticorp. https://aistudio.baidu.com/competition/detail/50/0/task-definition, 2020. Online; last updated in 2020.

[12] Qin Lu Minglei Li, Yunfei Long and Wenjie Li. "emotion corpus construction based on selection from hashtags.". In *International Conference on Language Resources and Evaluation (LREC), Portorož, Slovenia*, 2016.