

基于情感分析技术的餐馆评分系统

小组成员及分工

分数占比	班级	姓名	学号	分工
50%	2020219111	刘帅	2020212267	系统流程设计、环境搭建、chatGLM-6B及UIE模型部署
20%	2020219111	史桂宇	2020212268	前期调研、报告撰写、ppt设计
30%	2020219112	张骁	2020212276	数据集处理、模型蒸馏、综合评分等部分编写

一、项目背景及意义

1.1 项目背景

随着人们生活水平的提高和消费观念的改变，越来越多的人选择在外就餐。但是，面对餐饮市场的繁荣和多样化，消费者往往难以选择到自己满意的餐馆。因此，我们基于uie通用情感模型，开发一个餐馆评分系统，可以为消费者提供方便、快捷、准确的餐馆评价信息，帮助消费者更好地选择餐馆，同时也可以为餐馆提供改进和优化的方向。

1.2 项目意义

1. 提供准确、及时的餐馆评分信息，为消费者提供帮助，提高消费者的选择效率和消费体验。
2. 帮助餐馆了解消费者的评价和需求，为餐馆优化服务和改进经营提供参考。
3. 让消费者和餐馆之间的交流更加直接和有效，促进消费者和餐馆之间的互动和沟通。
4. 为餐饮市场的健康发展提供参考和指导，促进餐饮市场的规范化和优化。

二、系统架构

项目运行环境：

```
1 nvcc: NVIDIA (R) Cuda compiler driverCopyright (c) 2005-2021 NVIDIA CorporationBuilt on Sun Mar 21 19:15:46 PDT 2021Cuda compilation tools, release 11.3, V11.3.58Build cuda 11.3.r11.3/compiler.29745058 0
2
3 paddle version: 2.4.2
4 paddlenlp version: 2.5.2.post
5 torch version: 1.10.1
```

2.1 系统实现的功能

- 1、用户输入对某个餐馆的评价，使用UIE模型进行属性级情感分析，并由系统生成星级评分（1~5）。
- 2、利用ChatGLM-6B模仿人类给出关于餐厅的评价，从而不断扩充数据，打造数据闭环。
- 3、针对用户的评价，ChatGLM-6B智能助手会根据用户的评论给商家以改进方案，为Embodied AI的reasoning任务提供思路。
- 4、构建轻量化的ChatGLM微调模型，便于在移动端或机器人进行部署。

2.2 系统使用的自然语言技术

2.2.1 属性级情感分析

属性级情感分析用于针对不同属性（aspect）进行情感分类（正向、中性、负向）并最终用于获取评论的评分。

可以用于属性级情感分析的模型有LSTM、BiLSTM、CNN、BERT等。

1. LSTM

长短期记忆（Long short-term memory, LSTM）是一种特殊的RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，就是相比普通的RNN，多了一个门(gate)机制和细胞记忆单元(cell-state)用来存储，所以LSTM能够在更长的序列中有更好的表现。

He R, Lee W S, Ng H T, et al. Exploiting document knowledge for aspect-level sentiment classification[J]. arXiv preprint arXiv:1806.04346, 2018. <https://aclanthology.org/P18-2092/>

2. BiLSTM

虽然LSTM解决了RNN会发生梯度消失或者梯度爆炸的问题,但是LSTM只能学习当前词之前的信息,不能利用当前词之后的信息,由于一个词的语义不仅与之前的历史信息有关,还与当前词之后的信息也有着密切关系, BiLSTM 代替LSTM既解决了梯度消失或者梯度爆炸的问题,又能充分考虑当前词的上下文信息。

BiLSTM结合了序列起点移动的一个循环神经网络和一个从序列末尾向序列起点移动的循环神经网络。而作为循环神经网络的一种拓展，LSTM 自然也可以结合一个逆向的序列，组成双向长短时记忆网络Bi-LSTM。

3. CNN

CNN本质上是一个多层感知机，其被广为使用的原因关键在于它所采用的局部感受野和共享权值的方式，一方面减少了权值的数量使得网络易于优化，另一方面降低了过拟合的风险。其是神经网络中的一种，它的权值共享网络结构使之更类似于生物神经网络，降低了网络模型的复杂度，减少了权值的数量。CNN还有一个重要思想就是池化，池化层通常接在卷积层后面，引入它的目的就是为了简化卷积层的输出。

LI Yang, DONG Hongbin. Text sentiment analysis based on feature fusion of convolution neural network and bidirectional long short-term memory network[J]. Journal of Computer Applications, 2018, 38(11): 3075-3080. <http://www.joca.cn/CN/Y2018/V38/I11/3075>

4. BERT

BERT可以通过无监督的学习海量文本构建出通用语言模型。训练好的预模型只需通过少量的有监督训练微调参数，就可以应用于特定自然语言任务。

[BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis](#)
(Xu et al., NAACL 2019)

与大模型UIE相比，这些传统模型存在一些缺点：

1. 无法捕捉长距离依赖性

处理长文本时可能存在困难，无法有效地捕捉长距离的依赖关系。相比之下，大模型UIE通过使用更复杂的注意力机制和更深层的网络结构，能够更好地捕捉文本中的长距离依赖关系。

2. 参数量限制

传统模型通常具有固定的网络结构和参数数量，不适用于处理大规模的数据集或长文本。相比之下，大模型UIE通常具有更大的模型容量和更多的参数，可以更好地处理大规模数据和复杂文本。

3. 难以处理语义理解

在语义理解方面存在一定的局限性。它们主要通过局部特征的组合或时序建模来获得文本的表示，但对于一些复杂的语义关系和推理任务，这些模型可能无法提供足够准确的表示。大模型UIE通过预训练和自监督学习等技术，可以学习到更丰富的语义表示，并在语义理解任务上表现更好。

4. 训练和调参复杂性

相对于传统模型，大模型UIE的训练和调参过程通常更加复杂和耗时。大模型UIE通常需要更多的计算资源和训练数据，并且需要进行更多的超参数调优和模型微调才能达到最佳性能。而传统模型如通常具有更简单的训练和调参过程。

UIE的优势：

1. 使用简单

用户可以使用自然语言自定义抽取目标，无需训练即可统一抽取输入文本中的对应信息。实现开箱即用，并满足各类信息抽取需求。

2. 降本增效

以往的信息抽取技术需要大量标注数据才能保证信息抽取的效果，为了提高开发过程中的开发效率，减少不必要的重复工作时间，开放域信息抽取可以实现零样本（zero-shot）或者少样本（few-shot）抽取，大幅度降低标注数据依赖，在降低成本的同时，还提升了效果。

3. 效果领先

开放域信息抽取在多种场景，多种任务上，均有不俗的表现。

2.2.2 知识蒸馏

知识蒸馏 (Knowledge Distillation) 是一种模型压缩技术，可以通过将一个大型模型的知识转移到一个小型模型中，来提高小型模型的性能和效率。在 NLP 领域中，知识蒸馏算法同样可以用于将一个大型预训练模型的知识转移给一个小型模型，以便在计算资源有限的情况下提高模型的效率。

以下是几种常用的知识蒸馏方法：

1. 蒸馏到浅层网络：这种方法的目标是将大型模型的知识蒸馏到浅层网络中。具体而言，可以通过使用大型模型的中间层输出作为小型模型的输入来实现知识蒸馏。这种方法通常可以使小型模型在保持较高准确率的同时，具有更快的推理速度。
2. 蒸馏到多个任务：这种方法的目标是将大型模型的知识蒸馏到能够处理多个任务的小型模型中。具体而言，可以通过使用大型模型对多个任务的预测结果作为小型模型的监督信号来实现知识蒸馏。这种方法通常可以使小型模型在多个任务上表现出色。
3. 蒸馏到生成模型：这种方法的目标是将大型模型的知识蒸馏到生成模型中。具体而言，可以通过使用大型模型生成的概率分布作为小型模型的目标分布来实现知识蒸馏。这种方法通常可以使小型模型在生成任务上表现出色。
4. 蒸馏到同构网络：这种方法的目标是将大型模型的知识蒸馏到与其同构的小型模型中。具体而言，可以通过使用大型模型和小型模型具有相同结构的中间层来实现知识蒸馏。这种方法通常可以使小型模型在保持较高准确率的同时，具有更小的模型大小。

在我们的系统中，使用了同构网络实现了模型的蒸馏。

2.2.3 文本生成

文本生成是一类 NLP 任务，旨在自动生成连续的文本序列，包括文本摘要、机器翻译、对话生成、语言模型等。有许多不同的方法可以用于生成连续的文本序列，包括基于规则、统计、Seq2Seq、自回归模型和 GAN 等方法。每种方法都有其优点和缺点，可以根据具体的任务和数据集选择最适合的方法。以下是几种常见的文本生成方法：

1. 基于规则的文本生成：这种方法使用一些预定义的规则来生成文本，例如模板填充或基于规则的翻译，因为其过于简单，因此只适用于特定类型的文本生成任务。
2. 基于统计的文本生成：这种方法使用统计模型来生成文本，例如 n-gram 语言模型或基于神经网络的语言模型，可以适用于各种文本生成任务。具体而言，这些模型会学习文本中的统计规律，然后使用这些规律来生成新的文本序列。
3. 基于序列到序列 (Seq2Seq) 模型的文本生成：这种方法使用 Seq2Seq 模型来学习输入序列和输出序列之间的映射关系，例如机器翻译任务。具体而言，Seq2Seq 模型由编码器和解码器两部分组成，编码器将输入序列编码成一个中间表示，解码器使用该中间表示生成输出序列。
4. 基于自回归模型的文本生成：这种方法使用自回归模型来生成文本序列，例如循环神经网络 (RNN) 或变压器 (Transformer) 等模型。具体而言，这些模型将输入序列依次输入，并且在每个时间步骤上生成一个新的单词或字符，然后将其作为下一个时间步骤的输入。

5. 基于生成对抗网络 (GAN) 的文本生成：这种方法使用 GAN 模型来生成文本，其中生成器模型学习生成假文本，而判别器模型学习区分真实文本和生成文本。具体而言，生成器模型生成假文本，判别器模型将其与真实文本进行比较并给出一个概率值，生成器模型通过优化目标来使得它生成的假文本更接近真实文本，而判别器模型则通过优化目标来使得它正确地区分真实文本和假文本。

在我们的系统中，**使用ChatGLM-6B模型进行文本生成，针对用户的不同评语以店家的身份进行回复，提出相关改进措施。**GPT（Generative Pre-trained Transformer）是一种基于 Transformer 模型的预训练语言模型，旨在学习自然语言的概率分布。使用 GPT 作为文本生成模型的优点包括：

1. 预训练模型的效果好：GPT 模型使用大规模的语料库进行预训练，可以学习到自然语言的一些基本规律和规范，因此在一些文本生成任务上表现出色。
2. 可以生成连贯的、自然的文本：GPT 模型是一种自回归模型，可以生成连贯的、自然的文本序列。在生成文本时，模型可以利用前面生成的文本作为上下文，从而产生更加合理的文本。
3. 可以生成多样化的文本：GPT 模型使用随机采样或者束搜索等方法来生成多个不同的文本序列，从而可以产生多样化的文本。
4. 可以通过微调进行迁移学习：GPT 模型可以在大规模的文本数据上进行预训练，然后通过微调在特定任务上进行迁移学习。这种方法可以显著降低训练时间和数据需求，并且可以在少量数据的情况下获得较好的效果。
5. 可以生成长文本：GPT 模型可以生成任意长度的文本，因此可以用于生成较长的文本序列，例如文章、小说等。

2.3 关于自动评分的实现逻辑

由于各种算法都不是很能直接获得符合实际的星级结果，因此对于最终星级的评价进行如下处理, 其中 s 是通过公式计算得到的分数, $Score$ 表示最终呈现的分数。

$$Score = \min\{5, \max\{1, \frac{\lfloor s * 2 + 1 \rfloor}{2}\}\}$$

2.3.1 使用简单算术获得用户评分

使用简单算术法获得用户平均分的方法为：

获取用户在每个情感属性上的极性，并赋予相应的分数（积极 5，中立 3，消极 1，未提及的属性不参与），然后取平均得到最终的分数。

$$s = \frac{n_{pos} * 5 + n_{neg} * 3 + n_{neu} * 1}{n_{pos} + n_{neg} + n_{neu}}$$

2.3.2 使用线性回归获得用户评分

根据评分数据和情感分析所得的情感极性，通过线性回归得出每个属性（aspect）的权重，对于新数据，提取情感分析维度后以如下公式获得最终评分。（ $\lfloor s \rfloor$ 表示结果对结果 s 向下取整）

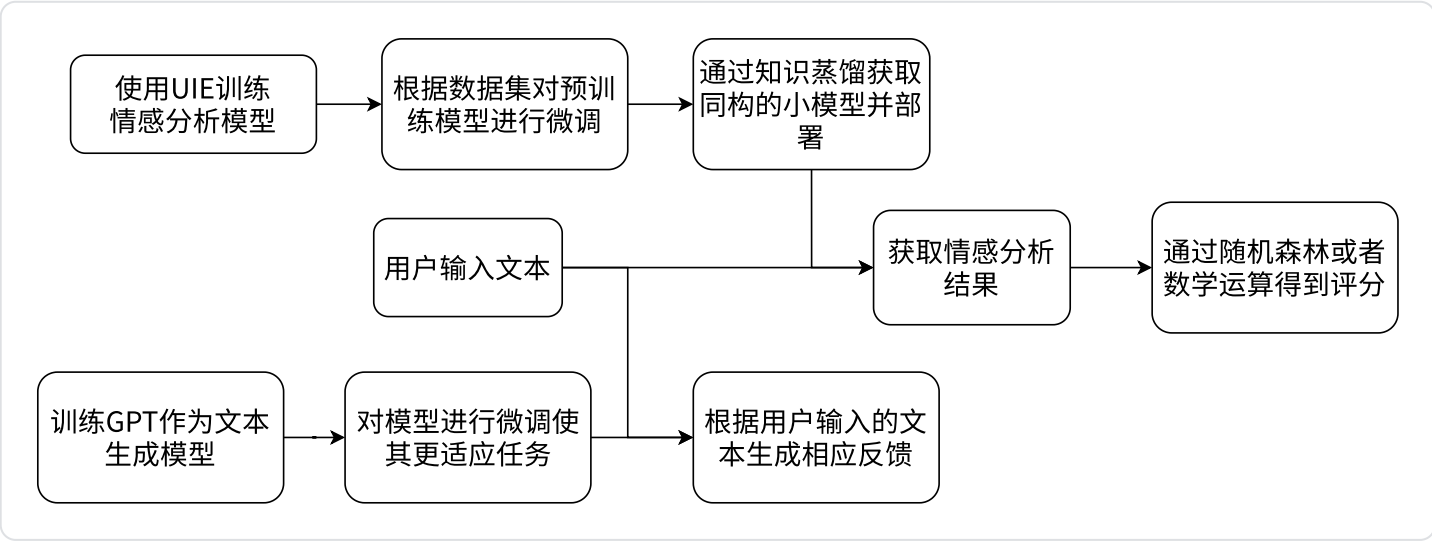
$$s = \sum_{i \in A} \omega_i \times f_i$$

其中 \mathcal{A} 是属性的集合， ω_i 表示属性 i 对应的权重， f_i 表示属性级情感分析中属性 i 的情感极性（ $\pm 1, 0$ ）。

$$f_i = \begin{cases} -2, & \text{表示属性未提及} \\ -1, & \text{属性}i\text{的情感极性为负向} \\ 0, & \text{属性}i\text{的情感极性为中性} \\ 1, & \text{属性}i\text{的情感极性为正向} \end{cases}$$

2.3 系统总体框架

系统整体功能的实现如下所示。



其中初始化涉及到的步骤包括

1. 将csv格式的数据集转化为json格式（在3.2会详细介绍）
2. 调用UIE预训练模型，对模型进行微调和评估
3. 使用同构模型对UIE模型进行知识蒸馏，得到用于部署的小模型
4. 调用GPT预训练模型，对模型进行微调和评估

用户使用时系统的步骤

1. 用户输入评价，由UIE模型进行情感分析并返回
2. 根据UIE模型的情感分析结果调用模型或使用算式计算得分
3. 调用GPT模型生成对用户评价的相应反馈
4. 用户可以使用GPT模型生成餐厅评价，此项也可以用于数据增广

三、数据集介绍

3.1 情感分析数据集

使用ASAP中文评论数据集：aspect的情感分析（<https://github.com/Meituan-Dianping/asap>）

ASAP (Aspect-based Sentiment Analysis & Rating Prediction的简写) 是一个大规模的中文餐馆评论数据集ASAP，包括来自中文一个领先的线上到线下 (O2O) 电子商务平台的46730条真实评论。除了五星评分外，每条评论都根据其对18个预先定义的aspect类 (如下表所示) 的情感倾向进行了人工标注。

Table 1: The full list of 18 aspect categories and definitions.

Aspect category	Definition	Aspect category	Definition
<i>Food#Taste</i> (口味)	Food taste	<i>Location#Easy_to_find</i> (是否容易寻找)	Whether the restaurant is easy to find
<i>Food#Appearance</i> (外观)	Food appearance	<i>Service#Queue</i> (排队时间)	Whether the queue time is acceptable
<i>Food#Portion</i> (分量)	Food portion	<i>Service#Hospitality</i> (服务人员态度)	Waiters/waitresses' attitude/hospitality
<i>Food#Recommend</i> (推荐程度)	Whether the food is worth being recommended	<i>Service#Parking</i> (停车方便)	Parking convenience
<i>Price#Level</i> (价格水平)	Price level	<i>Service#Timely</i> (点菜/上菜速度)	Order/Serving time
<i>Price#Cost_effective</i> (性价比)	Whether the restaurant is cost-effective	<i>Ambience#Decoration</i> (装修)	Decoration level
<i>Price#Discount</i> (折扣力度)	Discount strength	<i>Ambience#Noise</i> (嘈杂情况)	Whether the restaurant is noisy
<i>Location#Downtown</i> (位于商圈附近)	Whether the restaurant is located near downtown	<i>Ambience#Space</i> (就餐空间)	Dining Space and Seat Size
<i>Location#Transportation</i> (交通方便)	Convenient public transportation to the restaurant	<i>Ambience#Sanitary</i> (卫生情况)	Sanitary condition

在情感分析中，ASAP主要关注aspect类别分析，即只对给定的aspect进行情感倾向的判定。例如给定一个评论 "虽然鱼很好吃，但服务员太可怕了！"，ACSA任务的目的是推理出对aspect类别食品的情感倾向是积极（1）的，而对aspect类别服务的意见是消极（-1）的，此外还有表示情感倾向为中性的0。下图展示了一个具体的句子（Review）的属性级情感分析的标注（如右部所示）。

Table 3: A review example in ASAP, with overall star rating and aspect category sentiment polarity annotations.

Review	Rating	Aspect Category	Label	Aspect Category	Label
<p>With convenient traffic, the restaurant holds a high-end decoration, but quite noisy because a wedding ceremony was being held in the main hall. Impressed by its delicate decoration and grand appearance though, we had to wait for a while at the weekend time. However, considering its high price level, the taste is unexpected. We ordered the Kung Pao Prawn, the taste was acceptable and the serving size is enough, but the shrimp is not fresh. In terms of service, you could not expect too much due to the massive customers there. By the way, the free-served fruit cup was nice. Generally speaking, it was a typical wedding banquet restaurant rather than a comfortable place to date with friends.</p> <p>交通还挺方便的，环境看起来很高大上的样子，但是因为主厅在举办婚礼非常混乱，特别吵感觉，但是装修的还不错，感觉很精致的装修，门面很气派，周末去的时候还需要等位。味道的话我觉得还可以但是跟价格比起来就很一般了，性价比挺低的，为了去吃宫保虾球的，但是我觉得也就那样吧虾不是特别新鲜，不过虾球很大，味道还行。服务的话由于人很多所以也顾不过来上菜的速度不快，但是有送水果杯还挺好吃的。总之就是典型的婚宴餐厅不是适合普通朋友吃饭的地方了。</p>	3-Star	Location#Transportation (交通方便)	1	Price#Discount (折扣力度)	-
		Location#Downtown (位于商圈附近)	-	Ambience#Decoration (装修)	1
		Location#Easy_to_find (是否容易寻找)	-	Ambience#Noise (嘈杂情况)	-1
		Service#Queue (排队时间)	-	Ambience#Space (就餐空间)	1
		Service#Hospitality (服务人员态度)	-	Ambience#Sanitary (卫生情况)	-
		Service#Parking (停车方便)	-	Food#Portion (分量)	1
		Service#Timely (点菜/上菜速度)	-1	Food#Taste (口味)	1
		Price#Level (价格水平)	0	Food#Appearance (外观)	-
		Price#Cost_effective (性价比)	-1	Food#Recommend (推荐程度)	-

数据集整体分布情况如下。

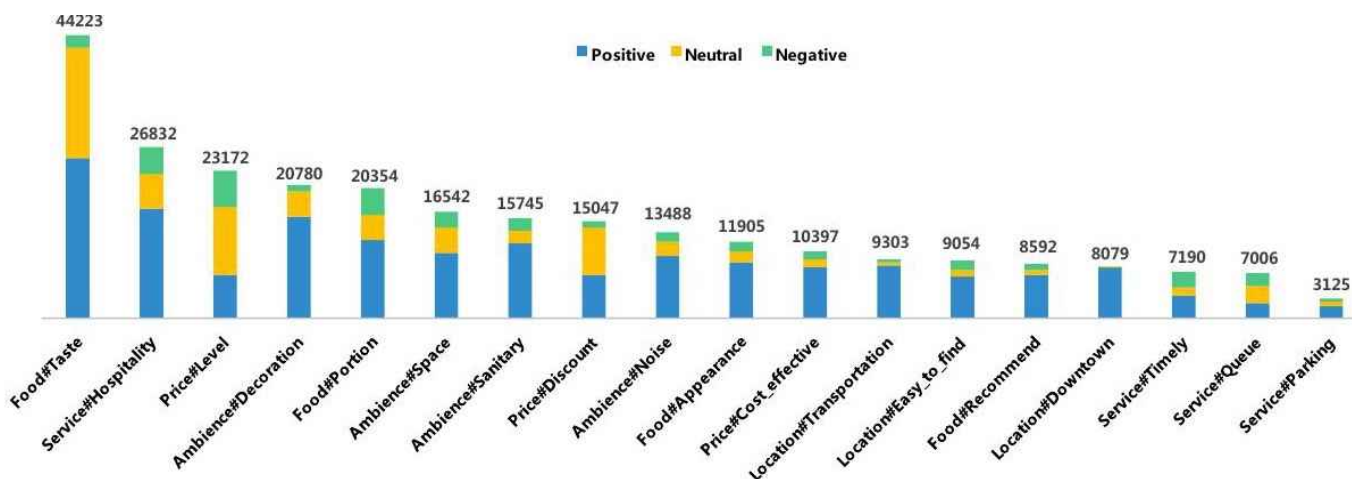


Figure 3: The distribution of 18 fine-grained aspect categories in ASAP.

这一部分数据集使用 `ASPADataset` 类来表示。

3.2 数据处理方法

自定义 `ASAPDataset` 来将csv转为json，以用于后续的训练。主要功能是将给定的 CSV 格式的数据集转换为 JSON 格式，并且将数据集分成训练集和验证集，以便于后续模型训练和评估。在 `ASAPDataset` 类的 `__init__()` 方法中，代码读取了给定的 CSV 文件，并将每条评论作为一个样本，将每个评价对象（如“口味”、“服务”等）转换为一个分类样本。对于每个评价对象，代码使用 `generate_cls_example()` 方法生成一个分类样本，其中包含了评论、评价对象和标签。最后，代码将转换后的所有样本保存到一个 JSON 文件中。

具体的实现过程如下：

1. 在 `__init__()` 方法中，代码使用 `csv.reader()` 函数读取 CSV 文件，并获取文件的标题和内容。

```
1 def __init__(self, data_file,rewrite=False):
2     super().__init__()
3     self.data = []
4     self.labels = []
5     self.scores = []
6     self.results={}
7
8     # json数据集保存路径
9     self.result_path = "./json"
10    self.data_name = data_file.split("/")[-1].split(".")[0]+".json"
11    self.save_json_path=os.path.join(self.result_path, self.data_name)
12    self.options=["正向","中性","负向","未提及"]
13    self.hash={'1':"正向",'0':"中性","-1":"负向","-2":"未提及"}
14    self.example=[]
```

2. 对于每一行数据，代码将其转换为一个字典类型的数据项，并将其保存到 `self.results` 字典中。其中，键为评论的 ID，值为一个包含评论内容和每个评价对象的字典。
3. 对于每个评价对象，代码使用 `generate_cls_example()` 方法生成一个分类样本。该方法将评价对象的标签（正向、中性、负向或未提及）随机分配到一组选项中，形成一个分类样本，其中包含了评论、评价对象和标签。

```
1 def generate_cls_example(self, text, label, prompt_prefix):
2     random.shuffle(self.options)
3     cls_options = ",".join(self.options)
4     prompt = prompt_prefix + "[" + cls_options + "]" #eg.口味纯正的情感倾向
5     result_list = []
6     example = {"content": text, "result_list": result_list, "prompt": prompt}
7     start = prompt.rfind(label) - len(prompt) - 1
8     end = start + len(label)
9     result = {"text": label, "start": start, "end": end}
10    example["result_list"].append(result)
```

4. 生成的每个分类样本保存在 `self.example` 列表中，该列表最终被随机打乱。
5. 最后，代码将所有的分类样本保存到一个 JSON 文件中，使用 `json.dumps()` 函数将字典转换为 JSON 格式的字符串，并使用 `write()` 方法将字符串写入文件中。在写入 JSON 文件时，代码使用 `a+` 模式打开文件，以便于写入多个 JSON 对象。

```

1     def splitdata(self,prot,data):
2         random.shuffle(data)
3         trainlen=int(len(data)*prot[0])
4         devlen=int(len(data)*prot[1])
5         traindata=data[:trainlen]
6         devdata=data[trainlen:]
7         return traindata,devdata
8     def writejson(self,example,jsonpath):
9         with open(jsonpath, 'a+', encoding='utf-8') as f:
10             f.write(json.dumps(example, ensure_ascii=False) + "\n")

```

在原csv文件中，每一条大致如下。(第一行是标题)，除了id,review之外，其余的是对餐馆的评价，star是用户打的星级，后面的是各个情感属性的极性，如上所述。

```

1 id,review,star,Location#Transportation,Location#Downtown,Location#Easy_to_find,
  Service#Queue,Service#Hospitality,Service#Parking,Service#Timely,Price#Level,Pr
  ice#Cost_effective,Price#Discount,Ambience#Decoration,Ambience#Noise,Ambience#S
  pace,Ambience#Sanitary,Food#Portion,Food#Taste,Food#Appearance,Food#Recommend
2 46277,状元楼饭店第一次去，因为地理位置优越：在宁波市和义大道高、大、上，里面装修中式，菜是
  地道的宁波菜，口味纯正，醉泥螺特棒，吃到了小时候的味道，因为去了晚了，在大堂等了一会儿，期
  间有茶水喝、服务员还与用户聊天，到了就餐时生意太好，服务员都是小跑状，服务态度绝对不提速，
  样样都服务到位，点酒水还耐心的与我们解释，就这样绝对要夸一夸，特别是彭新星、洪继华（看服务
  牌才知道名字）也给我们宁波市形象增色，状元楼是宁波的一扇窗口，服务员的素质更体现我们宁波人
  的精神面貌。赞一个,5.0,1,1,1,-2,1,-2,-2,-2,-2,-2,1,-2,-2,-2,-2,1,-2,-2

```

转为json格式之后，每条数据如下，只包含一个评论的某个评价维度的情感极性。

```

1 {
2     "content": "单位公司聚会，选择中午来中茵皇冠吃自助，高大上的酒店啊，我们浩浩荡荡三十人来到餐厅，到的时候已经十二点了，用餐的人真不少，挑了个座位放下东西，就拿了盘子去挑吃的，毕竟是中午，海鲜刺身不是很多，就吃了点虾和贝壳，热菜拿了龙利鱼，咖喱鸡，墨鱼仔等等，羊排没吃到，等的人太多了，每次都一抢而空，点心还不错，有一个不知道叫什么名字，里面是满满的虾，超级好吃，饮料还不错，现榨的，哈根达斯貌似一向很受欢迎，总体来说不错",

```

```
3     "result_list": [  
4         {  
5             "text": "正向",  
6             "start": -7,  
7             "end": -5  
8         }  
9     ],  
10    "prompt": "口味的情感倾向[负向,未提及,正向,中性]"  
11 }
```

四、UIE模型详细说明

4.1 模型介绍

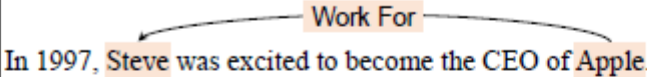
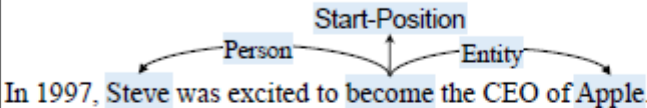
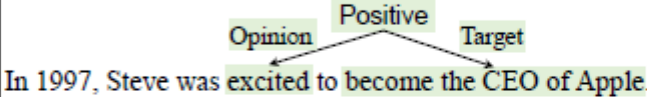
4.1.1 摘要

信息提取的问题在于目标多变性、结构异质性和需求特定模式。UIE框架通过生成模型统一处理不同的IE任务，自适应地生成目标结构，并从不同的知识来源协同学习。通过大规模预训练文本到结构模型，UIE在4个IE任务、13个数据集和各种设置上实现了最先进的性能，验证了其有效性、普适性和可移植性。

4.1.2 介绍

信息抽取(Information extraction, IE)旨在从非结构化文本中识别和构建用户指定的信息。

从根本上说，所有IE任务都可以建模为文本到结构的转换，使用不同的任务对应不同的结构。如图1所示，实体是命名跨度结构，事件是模式定义的记录结构。

Task	Schema	Instance								
Entity	PER: _ ORG: _	In 1997, Steve was excited to become the CEO of Apple .								
Relation	(_, Work for, _)	In 1997, Steve was excited to become the CEO of Apple . 								
Event	<table border="1"><thead><tr><th>Type</th><th>Start Position</th></tr></thead><tbody><tr><td>employee</td><td></td></tr><tr><td>employer</td><td></td></tr><tr><td>...</td><td></td></tr></tbody></table>	Type	Start Position	employee		employer		...		In 1997, Steve was excited to become the CEO of Apple . 
Type	Start Position									
employee										
employer										
...										
Sentiment	Positive { Opinion: _; Target: _ }	In 1997, Steve was excited to become the CEO of Apple. 								

(a) Task-specialized IE

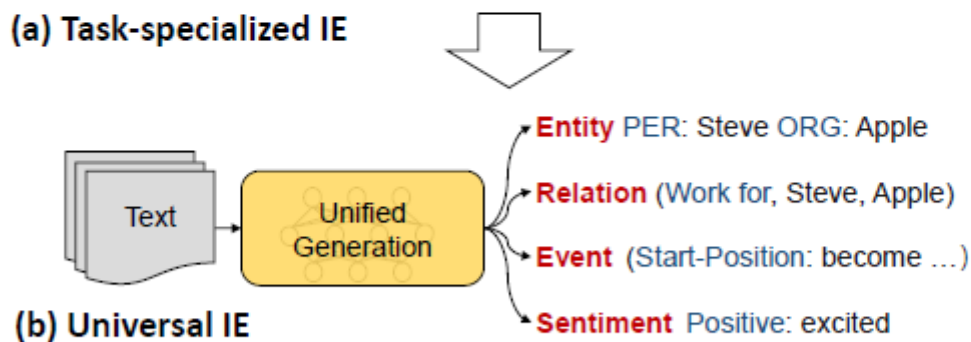


Figure 1: From (a) Task-specialized IE: different tasks, different structures, different schemas to (b) Universal IE: unified modeling via structure generation.

4.1.3 用于通用信息提取的统一结构生成

信息提取任务可以表述为文本到结构的问题，其中不同的IE任务对应不同的结构。本文旨在通过一个统一的框架对不同IE任务的文本到结构的转换进行统一建模，即不同的结构转换将在一个通用的模型中共享相同的底层操作和不同的转换能力。形式上，给定特定的预定义模式s和文本x，通用IE模型需要生成一个结构，该结构包含模式s所指示的文本x中所需的结构信息。

首先设计结构化抽取语言(structured extraction language, SEL)来统一编码异构抽取结构，即将实体、关系、事件编码为统一的表示。然后，我们描述了结构模式指导器(SSi)，这是一种基于模式的提示机制，它控制UIE模型，为不同的提取设置识别哪些、关联哪些以及生成哪些。

输出格式:SEL (structured extraction language)

作者认为这些抽取任务都可以由2种原子操作组合而成:

1. **Spotting**:信息点类别，把信息所在的位置找出来，比如实体类型;
 2. **Associating**:利用信息点关联类别把点连接起来;信息点关联类别，如关系类型、事件论元类型
- 不同IE任务都可以定义为文本到结构的生成。

生成的结构由抽取结构和需求 Schema 决定：

1. Entity :实体类型 + Span
2. Relation :关系类型/论元类型 + Triple
3. Event :事件类型/论元类型 + Frame

作者定义了一种称为 Structured Extraction Language(SEL) 格式的任务输出样式:

```
1 (
2   (Spot Name: Info Span
3     (Asso Name: Info Span)
4     (Asso Name: Info Span)
5   )
6 )
```

其中：

1. Spot Name :信息点类别，如实体类型；
2. Association Name (asso/asso) :信息点关联类别，如关系类型、事件论元类型；
3. Info Span :信息点所对应的文本片段

比如对于 steve became CEO of Apple in 1997. 这句话；

如果做NER，可以得到以下结果：

```
1 (
2   (person: Steve)
3   (organization: Apple)
4   (time: 1997)
5 )
```

如果做关系抽取，可以得到以下结果，其中的 work for 就是关系类别：

```
1 (
2   (person: Steve
3     (work for: Apple)
4     (organization: Apple)
5   )
```

如果做事件抽取，可以得到以下结果，其中的 start-position 可以理解为事件触发词，或者事件类别；employee 、 employer 、 time 则是事件中的角色或属性：


```
1 (  
2   (start-position: became  
3     (employee: Steve)  
4     (employer: Apple)  
5     (time: 1997)  
6   )  
7 )
```

一些示例：

Relation	CoNLL04	((location: Rome (located in: Lazio)) (location: Lazio) (location: Naples (located in: Campania)) (location: Campania))
Event	ACE05-Evt	((transport: heading (artifact: family) (destination: new hampshire) (origin: lakeland) (vehicle: plane)))
Sentiment	14/15/16-res	((aspect: staff (negative: horrible)) (opinion: horrible))

借助 SEL，就可以把各种抽取任务都统一到相同的表达框架

SEL具有以下优点：

- 1. 统一编码不同的IE结构，因此不同的IE任务可以被建模为相同的文本到结构的生成过程；
- 2. 有效地表示同一结构中句子的所有提取结果，从而可以自然地进行联合提取；
- 3. 生成的输出结构非常紧凑，大大降低了解码的复杂性。

输入格式:SSI ((structural schema instructor)

作者使用 `Structural Schema Instructor (SSI)`，把要抽取的 `Schema` 变成一串字符串，然后追加到待抽取的文本前面，一起输入 `Encoder`，`Schema` 信息转换为生成过程的提示语,指导UIE模型生成正确的信息,作为一个提示语，告诉模型从原始文本中定位正确的信息

结构化模式前与待抽取的文本一同输入序列到结构生成模型，用于分不同的抽取任务。基线模型使用特殊字符 `[spot]`、`[asoc]` 来织结构化模式前缀，`[spot]` 对应 SEL 中的 `SpotName` 类别，`[asoc]` 对应 `AssoName`。

不同任务的形式是：

1. 实体抽取: [spot] 实体类别 [text], 如 <spot> location <spot> organization <spot> person <spot> vehicle <spot> weapon
2. 关系抽取: [spot] 实体类别 [asoc] 关系类别 [text], 如 <spot> location <spot> organization <spot> other <spot> people <asoc> kill <asoc> live in <asoc> located in <asoc> organization in <asoc> work for
3. 事件抽取: [spot] 事件类别 [asoc] 论元类别 [text], 如 <spot> sue <spot> transfer money <spot> transfer ownership <spot> transport <spot> trial hearing <asoc> adjudicator <asoc> agent <asoc> artifact <asoc> attacker
4. 情感抽取: [spot] 评价维度 [asoc] 观点类别 [text], 如 <spot> aspect <spot> opinion <asoc> negative <asoc> neutral <asoc> positive

以夺冠事件为例, 其对应的SSI为 [spot] 夺冠 [asoc] 夺事件 [asoc] 冠军 [asoc] 夺冠赛事 [text] 2月8日上午北京冬奥会自由...。

结构生成与UIE

根据给定的SSI s 和文本 x 作为输入, UIE通过生成线性化的SEL来提取目标信息。我们使用编码器-解码器风格的架构来表述这个从文本到SEL生成的过程。给定原始文本序列 x 和模式指示器 s , UIE首先计算每个标记的隐藏表示:

$$H = [s_1, \dots, s_{|s|}, x_1, \dots, x_{|x|}]$$

其中 $H = \text{Encoder}(s_1, \dots, s_{|s|}, x_1, \dots, x_{|x|})$, $\text{Encoder}(\cdot)$ 是一个Transformer编码器。

然后, UIE将以自回归的方式将输入文本解码为线性化的SEL。在解码的第 i 步中, UIE根据以下方式生成SEL序列中的第 i 个标记 y_i 和解码器状态 h_{di} :

$$y_i; h_{di} = \text{Decoder}([H; h_{d1}; \dots; h_{di-1}])$$

其中 $\text{Decoder}(\cdot)$ 是一个Transformer解码器, 用于预测标记 y_i 的条件概率 $p(y_i | y_{<i}; x; s)$ 。最后, 当输出结束符号 $\langle \text{eos} \rangle$ 时, $\text{Decoder}(\cdot)$ 完成预测, 然后将我们预测的SEL表达式转换为提取的信息记录。

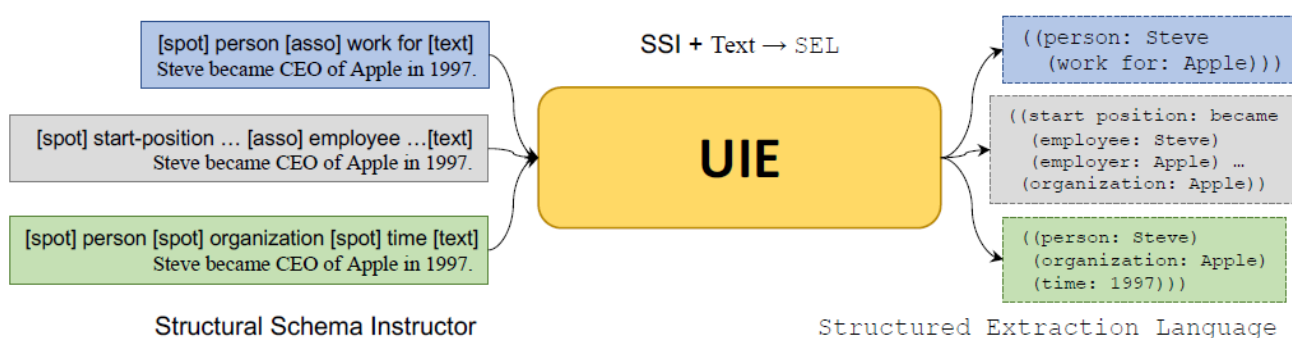


Figure 3: The overall framework of UIE.

4.1.4 模型预训练

预训练语料库构建

1. **Dpair** 是文本结构并行数据，其中每个实例是一个并行对(标记序列，结构化记录y)。来源是将维基数据与英文维基百科对齐来收集大规模并行级别的文本结构，对 **Dpair** 用于预训练UIE的文本到结构转换能力
2. **Drecord** 是结构化数据集，其中每个实例都是结构化记录y。从ConceptNet (Speer et al., 2017)和维基数据收集结构化记录，**Drecord** 用于预训练UIE的结构解码能力
3. **Dtext** 是非结构化文本数据集，使用英语维基百科中的所有纯文本。利用 **Dtext** 对UIE的语义编码能力进行预训练。

预训练

使用上述预训练数据集使用三个序列生成任务预训练UIE。

目标函数中包含了3部分。

1. 使用Dpair的文本到结构预训练。

为了获取基本的文本到结构映射能力，我们使用 $D_{\text{pair}} = \{(x, y)\}$ 对UIE进行预训练。

对于每个并行对 (x, y) ，我们提取记录y中的点类型 $ss+$ 和关联类型 $sa+$ 作为正模式 $s+ = ss+ \cup sa+$ 。然而，我们发现，如果我们只给UIE提供一个积极的模式，它只会简单地记住预训练数据中的三元组。为了学习一般的映射能力，我们还自动为每对构建负模式，即，我们首先对负点 $ss-$ 和负作为关联集 $sa-$ 进行采样，然后连接元模式 $s_{\text{meta}} = s+ \cup ss- \cup sa-$ ，并构建最终的抽取目标。

训练 Encoder-Decoder结构的目标是：

$$L_{\text{Pair}} = \sum_{(x, y) \in D_{\text{pair}}} -\log p(y|x, s_{\text{meta}}; \theta_e, \theta_d)$$

其中 θ_e 和 θ_d 分别是编码器和解码器的参数

2. 结构生成与Drecord预训练

训练 Drecoer，避免Drecoer解码出格式不对的结果。

为了预训练生成由SEL和模式定义的有效结构的能力，在Drecord上预训练UIE。将UIE解码器作为结构化语言模型进行预训练，其中Drecord中的每条记录都是SEL的表达式：

$$L_{\text{Record}} = \sum_{y \in D_{\text{record}}} -\log p(y_i|y_{<i}; \theta_d)$$

3. 使用Dtext改进语义表示。

在文本到结构的预训练过程中，我们还使用屏蔽语言模型任务在Dtext上持续预训练UIE，以改进UIE的语义表示。具体来说，我们在预训练阶段添加了基于跨度腐败的掩码语言建模目标：

$$L_{\text{Text}} = \sum_{x \in D_{\text{text}}} -\log p(x_{II}|x_I; \theta_e, \theta_d)$$

其中xi是损坏的源文本，xii是损坏的目标跨度。发现这种预训练可以有效地减轻令牌语义的灾难性遗忘，特别是在SPOTNAME和ASSONAME令牌上。

4. 最终预训练标准

使用T5-v1.1-base和T5-v1.1-large 初始化UIE -base 和UIE -large 。

Hyper-parameter	UIE-base	UIE-large
# Layers of Encoder	12	24
# Layers of Decoder	12	24
Hidden Dimension	768	1,024
FF hidden size	2,048	2,816
Layer Normalize ϵ	1e-6	1e-6
# Attention head	12	16
Attention head size	64	64

Table 7: Model architectures.

最终的目标是上述任务的组合:

$$L = L_{\text{Pair}} + L_{\text{Record}} + L_{\text{Text}}$$

我们将所有预训练数据统一表示为三元组。对于Dtext中的文本数据(x)，我们构建一个三元组(None, xi, xii)，其中xi是损坏的源文本，i0是损坏的跨度。对于Dpair中的文本记录数据(x, y)，我们通过采样每个文本记录对的元模式s来构造(s, x, y)。对于Drecord中的记录数据(y)，我们取(None, None, y)作为输入三元组。

4.2 模型部署

由于UIE项目是基于PaddleNLP框架进行构建的，因此我们需要配置paddlepaddle及paddlenlp的框架。

```
(paddle) root@LUCAS-DEV-b8d59a:/mnt/ve_share/liushuai/PaddleNLP# nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Mon_Nov_30_19:08:53_PST_2020
Cuda compilation tools, release 11.2, V11.2.67
Build cuda_11.2.r11.2/compiler.29373293_0
(paddle) root@LUCAS-DEV-b8d59a:/mnt/ve_share/liushuai/PaddleNLP#
```

查看/usr/local/目录，包含11.x三个cuda版本，首先在paddle官网安装对应版本的paddlepaddle框架

```
1 python -m pip install paddlepaddle-gpu==0.0.0.post112 -f https://www.paddlepaddle.org.cn/en/
```

安装paddlenlp框架

```
1 pip install --upgrade paddlenlp>=2.0.0rc -i https://pypi.org/simple
```

或采用python程序的安装方法，首先拉取paddleNLP的项目库

```
1 git@github.com:PaddlePaddle/PaddleNLP.git
2 cd PaddleNLP
```

之后执行 `python setup.py` 进行paddleNLP依赖的安装。

4.3 模型训练

4.3.1 训练数据

文件保存在 `unified_sentiment_extraction/Data/json` ,其中包括训练集 (`train.json`)、验证集(`dev.json`)和测试集(`test.json`)及其样例 (`train_sample.json`, `dev_sample.json`, `test_sample.json`)。(由csv格式转化为json格式的过程如3.2所述)

数据集中的其中一条如下所示，每一条是一个字典类型的变量

- `content` 是用户评论的内容。
- `prompt` 是评价的指标，如下示例中评价的维度是“口感”。
- `result_list` 也是一个字典，`text` 是 [负向,未提及,正向,中性] 其中的一个，因为这四个极性的顺序是随机排列的，所以增加了 `start` 和 `end` 表示起始和结束的位置。

```
1 {
2     "content": "单位公司聚会，选择中午来中茵皇冠吃自助，高大上的酒店啊，我们浩浩荡荡三十人来到餐厅，到的时候已经十二点了，用餐的人真不少，挑了个座位放下东西，就拿了盘子去挑吃的，毕竟是中午，海鲜刺身不是很多，就吃了点虾和贝壳，热菜拿了龙利鱼，咖喱鸡，墨鱼仔等等，羊排没吃到，等的人太多了，每次都一抢而空，点心还不错，有一个不知道叫什么名字，里面是满满的虾，超级好吃，饮料还不错，现榨的，哈根达斯貌似一向很受欢迎，总体来说不错",
3     "result_list": [
4         {
5             "text": "正向",
6             "start": -7,
7             "end": -5
8         }
9     ],
10    "prompt": "口味的情感倾向[负向,未提及,正向,中性]"
11 }
```


4.3.2 训练使用的依赖

```
1 import argparse # 用于解析命令行参数
2 import os # 执行文件操作
3 import time # 计时工具
4 from functools import partial # 部分应用函数
5 import sys; # 用于添加环境变量
6 sys.path.append("/mnt/ve_share/liushuai/PaddleNLP/applications/sentiment_analysis/unified_sentiment_extraction")
7
8 from evaluate import evaluate # 用于模型训练过程中评估
9 from utils import convert_example, create_data_loader, reader, set_seed # 一些实用工具
10
11 import paddle # 深度学习框架
12 from paddlenlp.datasets import load_dataset # 加载数据集
13 from paddlenlp.metrics import SpanEvaluator # 用于评估信息的
14 from paddlenlp.transformers import UIE, AutoTokenizer # 调用UIE模型及Tokenizer类
15 from paddlenlp.utils.log import logger # 用于记录日志
```

其中，`evaluate`和`utils`是根据具体任务编写的用于训练和测试的库。

- `paddlenlp.metrics.SpanEvaluator` 是PaddlePaddle 框架中用于评估文本序列标注任务的评价指标，例如命名实体识别（Named Entity Recognition，简称 NER）、词性标注（Part-of-Speech Tagging，简称 POS）等任务。该评价指标的实现基于准确率（Accuracy）和 F1 值（F1-score），适用于不同类型的序列标注任务。
- 在 `evaluate.py` 中，调用了 `evaluate` 函数，函数的主要步骤为作用是在数据加载器上循环迭代，每次迭代读取一个 batch 的数据，并根据模型疏忽和正式标签计算 Precision、Recall 和 F1 值，最后累加得到模型的评价指标（Precision、Recall 和 F1 值）。
- 在 `utils.py` 中引入了四个工具：`convert_example`，`create_data_loader`，`reader`，`set_seed`。
 - `set_seed` 用于生成随机种子。在训练神经网络时，使用随机梯度下降等优化算法进行参数更新时，每个 batch 的数据和初始参数的随机初始化都会对训练结果产生影响。在某些情况下，为了保证实验结果的可重复性，我们需要控制这些随机因素，即固定随机数种子，使得每次运行时使用的随机数都是确定的。
 - `convert_example` 的作用是将输入的原始样本数据转换为模型能够接受的格式，并返回转换后的数据。具体来说，该函数接受一个原始样本数据 `example`、一个 `Tokenizer` `tokenizer` 和一个最大序列长度 `max_seq_len`，并返回一个由 6 个 Numpy 数组组成的元组，分别表示输入的 token ID、token 类型 ID、位置 ID、attention mask、实体起始位置的标签和实体结束位置的标签。

- `create_data_loader` 的作用是创建一个数据加载器（`dataloader`），用于将数据集分批加载到模型中进行训练或验证。具体来说，该函数接受一个数据集 `dataset`、一个模式 `mode`、一个批量大小 `batch_size` 和一个转换函数 `trans_fn`，并返回一个数据加载器 `dataloader`。
- `reader` 的作用是读取一个包含多个 JSON 格式样本的文件，并按照一定规则将样本进行切分，从而生成一个可迭代的样本数据集。具体来说，该函数接受一个数据文件路径 `data_path` 和一个最大序列长度 `max_seq_len`，并返回一个生成器对象，该对象可以遍历数据文件中的所有样本数据。

4.3.3 训练过程

训练过程主要分为以下几个步骤：

1. 设置设备、分布式环境和种子：根据命令行参数，设置使用的设备，初始化分布式环境，设置随机数种子。
2. 加载 Tokenizer 和模型：使用 `AutoTokenizer.from_pretrained` 和 `UIE.from_pretrained` 函数分别加载自动选择的 Tokenizer 和预训练的 UIE 模型。
3. 加载数据集和数据加载器：使用 `load_dataset` 函数加载训练集和验证集，然后使用 `create_data_loader` 函数创建数据加载器，将数据集转换为模型能够接受的格式。
4. 初始化优化器、损失函数和评估器：使用 `paddle.optimizer.AdamW` 函数初始化 AdamW 优化器；使用 `paddle.nn.BCELoss` 函数初始化二元交叉熵损失函数；使用 `SpanEvaluator` 类初始化评估器，用于评估信息抽取任务的性能。
5. 训练模型：使用双重循环迭代训练数据集。每次迭代从数据加载器中读取一个 batch 的数据，并将其输入到模型中计算输出。然后计算模型输出与真实标签之间的损失，并反向传播更新参数。同时记录损失值、全局步数和训练时间，并每隔一定步数输出训练日志。在每个 epoch 结束时，使用验证集评估模型性能，并记录最佳的 F1 值和对应的模型参数。最后将训练日志和最佳模型保存到指定的文件路径

4.3.4 训练结果

在四个不同参数量大模型微调的训练过程中，我们选用了 **batchsize=16, learning_rate=1e-5, tokenizer 长度为 512, epoch=5** 的统一超参设置，并在每个 epoch 结束在相同的验证集进行验证。

model	Num_Params	Model Size (MB)	Precision	Recall	F1	ir
uie-senta-base	117946370	449.9	0.68177	0.71765	0.69925	3
uie-senta-medium	75429122	287.7	0.87157	0.84375	0.85743	2
uie-senta-mini	26950274	102.8	0.87205	0.84002	0.85573	2

uie-senta-micro	23401346	89.2	0.86310	0.84411	0.85350	2
uie-distill(from base)	23401346	89.2	0.76243	0.72383	0.74263	2

4.4 模型蒸馏

4.4.1 模型蒸馏的意义

几乎所有机器学习算法性能都可以通过一种非常简单的方法提升：在相同的数据上训练许多不同的模型，然后对它们的预测结果求平均值。但是，使用整个模型集合进行预测过于繁琐，而且如果各个模型都是庞大的神经网络，可能会过于计算密集，无法部署到大量用户。因此，模型压缩（在保证性能的前提下减少模型的参数量）成为了一个重要的问题。而”模型蒸馏“属于模型压缩的一种方法。模型蒸馏可以解决的大模型所面临与实际应用的冲突如下：

1. 推断速度慢
2. 对部署资源要求高(内存，显存等)
3. 在部署时，我们对延迟以及计算资源都有着严格的限制。

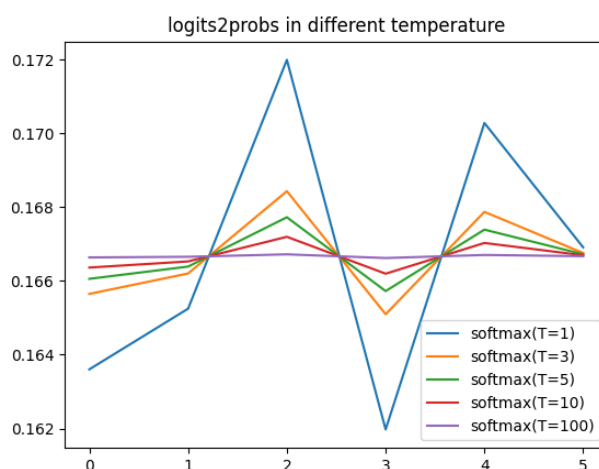
4.4.2 实现模型蒸馏

参考论文*Distilling the Knowledge in a Neural Network*。

对于神经网络来说，产生类别概率的方法是使用softmax输出层将logits中的元素(z_i)转化为一个与其他logits概率 (q_i)，公式如下。

$$q_i = \frac{\exp\{z_i/T\}}{\sum_j \exp\{z_j/T\}}$$

其中， T 表示温度，可以通过提升 T 来获取一个更加平滑（soft）的概率分布来增加对非正类信息的展示。下图是由 `logits = np.array([0.04, 0.05, 0.09, 0.03, 0.08, 0.06])` 在不同温度下进行softmax操作得到的数据可视化。分别选取了 $T=1, 3, 5, 10, 100$ 。可以观察到，在 $T=1$ 时数据比较符合 `one-hot`，而随着 T 的增大，非正类的数值增大，可以显示更多隐藏信息。但是当 T 超过一定范围（过大）时，就会消除每个类别之间的差异(每个类别概率相等，在图像上呈现一条直线)，因此在训练过程中应选取合适的温度。



在训练过程中，使用相同的高温来训练简单模型，但在训练完成后，简单模型使用温度为1的softmax函数进行预测。这种方法可以用来减小模型的大小和复杂度，同时保持模型的性能，从而更适合在资源受限的设备或实时应用中使用。在这种技术中，通过使用一个高温度的softmax函数来产生软目标分布，并使用这个分布将知识从一个复杂的模型传输到一个简单的模型上。

对训练得到的UIE模型规定蒸馏代码文件保存在

`unified_sentiment_extraction/zx_workplace/distillation.py` 中。代码使用知识蒸馏（Knowledge Distillation）来训练一个轻量级的UIE模型，以达到与原始模型相近的性能。

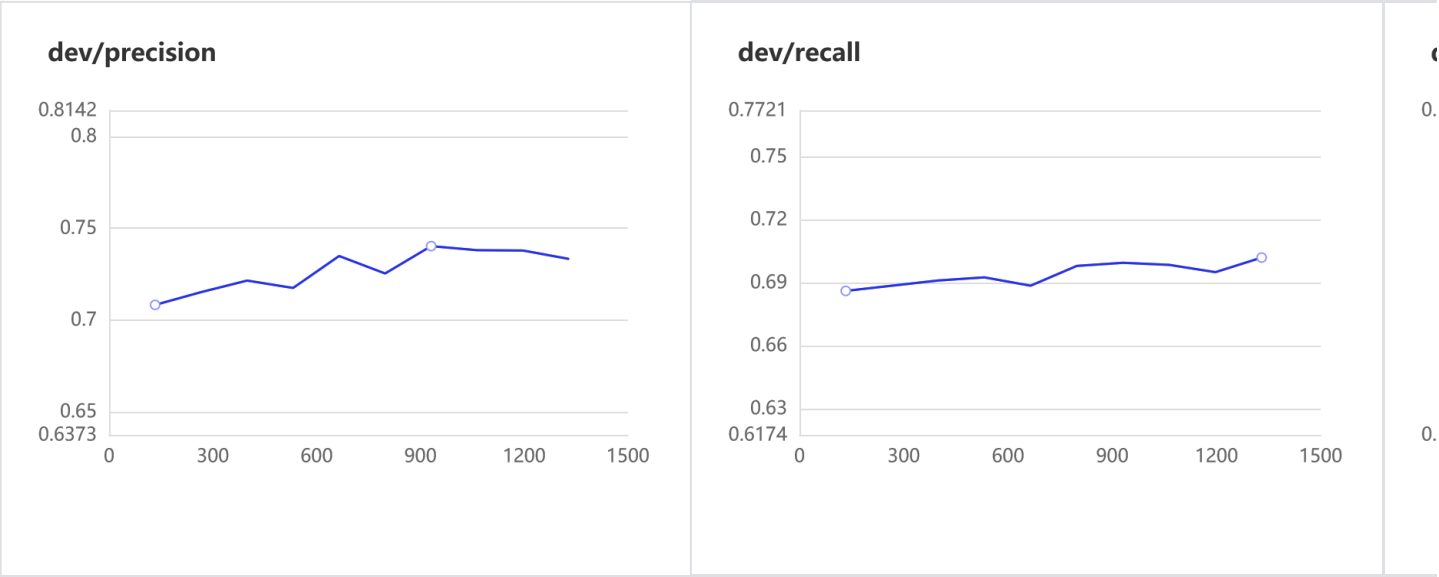
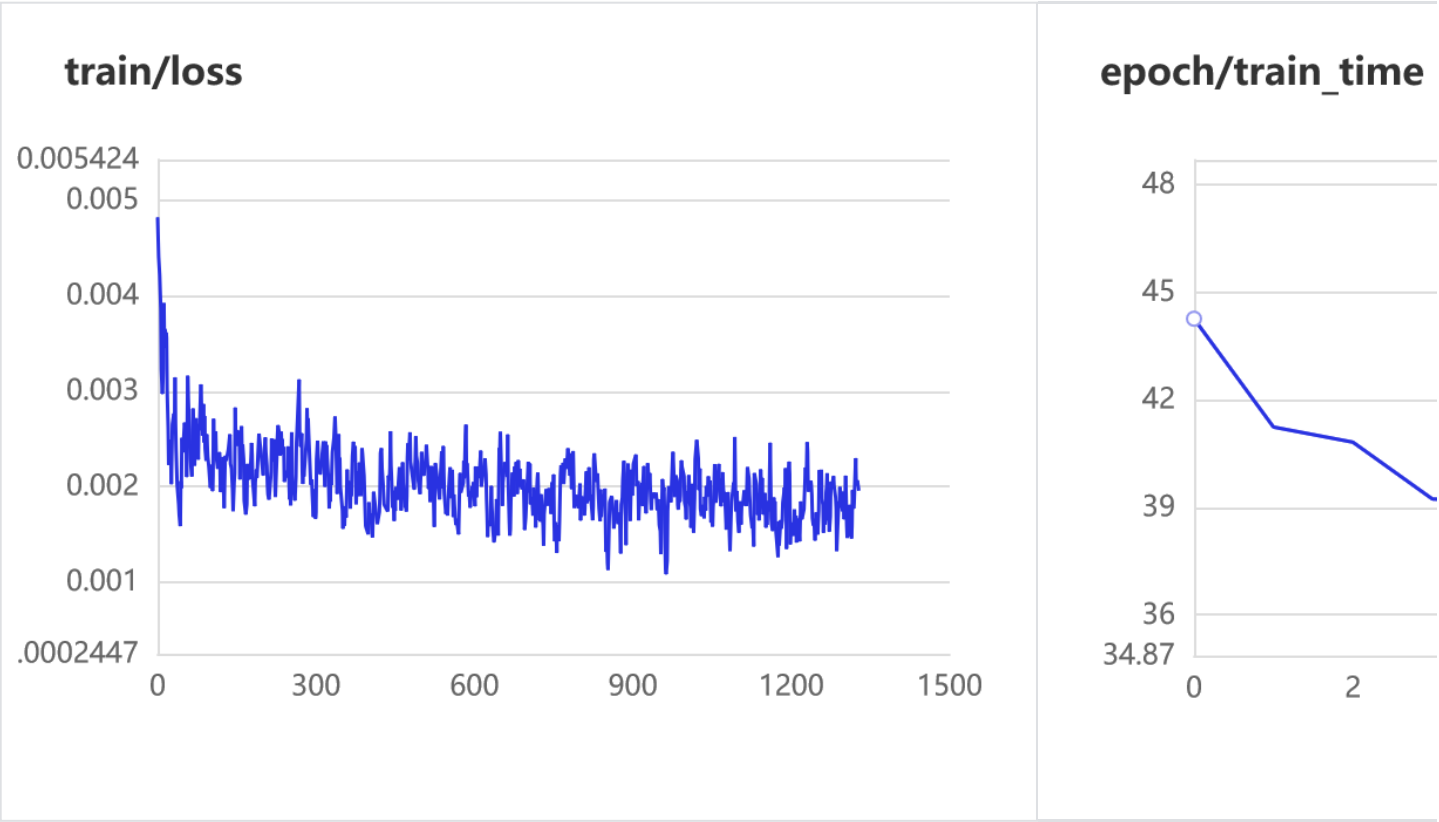
代码主要分为4部分：

```
1 # 1、导入必要的库
2 # 2、在一个epoch内根据硬标签和教师模型生成的软标签训练学生模型
3 def train_student(student_model, teacher_model, train_loader, optimizer, epoch,
4     """
5
6     :param student_model, teacher_model: student & teacher model
7     :param train_loader: training data loader
8     :param optimizer: optimizer
9     :epoch: training epoch
10    :criterion: used to calculate loss
11    :soft_weight: loss = soft_weight * soft_loss + (1 - soft_weight) * hard_loss
12    :global step: step of train
13    :rank: paddle.distributed.get_rank()
14    :return: global step
15    """
16 # 3、加载数据集，设置训练环境合并训练模型
17 def distill_knowledge():
18 # 4、设置参数以及运行（主函数）
```

1. 其中蒸馏所使用的数据是 `train_sample.json` , `dev_sample.json` , 并使用 `test.json` 评估，实现了数据蒸馏。
2. 生成软标签的方法是：首先暂停教师模型的梯度变化，然后对教师模型的预测结果进行温度缩放（应用softmax），然后再转为 `logits`，再以此为结果和学生模型的预测结果作交叉熵损失函数得到 `soft_loss`。
3. 训练过程中的损失函数表示为 `soft_weight * soft_loss + (1 - soft_weight) * hard_loss`。
4. 使用的优化器是 `paddle.optimizer.AdamW`，损失函数是 `paddle.nn.BCELoss`。

4.4.3 蒸馏的性能及结果

使用visuall(version 2.4.2)展示蒸馏的演示效果。训练epochs=10, 在训练过程中展示loss的变化，以及每一轮的训练时间如下。



epoch	Precision	Recall	F1	Time(s)
1	0.70814	0.68601	0.69690	44
2	0.71510	0.68849	0.70154	41
3	0.72139	0.69097	0.70585	40

4	0.71737	0.69246	0.70469	39
5	0.73478	0.68849	0.71088	39
6	0.72526	0.69792	0.71132	40
7	0.74016	0.69940	0.71920	38
8	0.73795	0.69841	0.71764	39
9	0.73776	0.69494	0.71571	39
10	0.73316	0.70188	0.71718	40
测试集表现	0.76243	0.72383	0.74263	\

4.4.4 蒸馏结果比较

我们以base模型作为teacher模型，用于生成供给uie-distill模型训练的软标签，在uie-distill上进行蒸馏。

在batchsize=16,learning_rate=1e-5,tokenizer长度为512，epoch=5统一超参设置的情况下，对训练结果进行比较：

可见，uie-distill在更短的时间内完成了训练，达到了比base更好的效果，同时，distill模型有着更小的参数量，更加轻量化。

epoch	Num_Params	Model Size (MB)	Precision	Recall	F1	ir
uie-senta-base	117946370	449.9	0.68177	0.71765	0.69925	3
uie-distill(from base)	23401346	89.2	0.76243	0.72383	0.74263	2

五、ChatGLM-6B 智能助手

参考网址<https://github.com/THUDM/ChatGLM-6B>

5.1 模型简介

目前，预训练模型的架构主要包括自编码模型（例如BERT）、自回归模型（例如GPT）和编码-解码模型（例如T5）。然而，没有一种预训练框架在所有三个主要类别的任务中表现最佳，这些任务包括自然语言理解（NLU）、

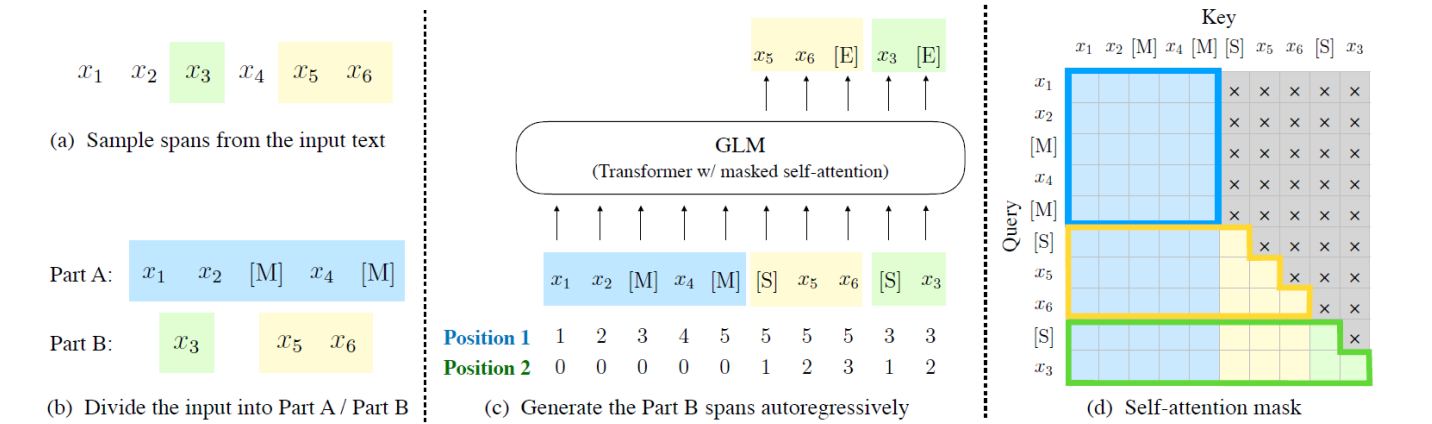
无条件生成（Unconditional Generation）和条件生成（Conditional Generation）。为了解决这个问题，我们提出了一种基于自回归空白填充的通用语言模型（GLM）。GLM通过添加二维位置编码和

允许任意顺序预测跨度来改进空白填充预训练，从而在NLU任务上表现优于BERT和T5。同时，GLM可以通过变化空白的数量和长度来预训练不同类型的任务。

ChatGLM-6B ChatGLM-6B 是一个开源的、支持中英双语的对话语言模型，基于 [General Language Model \(GLM\)](#) 架构，具有 62 亿参数。结合模型量化技术，用户可以在消费级的显卡上进行本地部署（INT4 量化级别下最低只需 6GB 显存）。ChatGLM-6B 使用了和 ChatGPT 相似的技术，针对中文问答和对话进行了优化。经过约 1T 标识符的中英双语训练，辅以监督微调、反馈自助、人类反馈强化学习等技术的加持，62 亿参数的 ChatGLM-6B 已经能生成相当符合人类偏好的回答。使用了基于 [P-Tuning v2](#) 的高效参数微调方法。

GLM预训练过程如下所示：

- （a）原始文本为 $[x_1; x_2; x_3; x_4; x_5; x_6]$ 。从中随机采样两个跨度 $[x_3]$ 和 $[x_5; x_6]$ 。
 - （b）在Part A中，将采样的跨度替换为 $[M]$ ，在Part B中打乱跨度的顺序。
 - （c）GLM自回归地生成Part B。每个跨度都以 $[S]$ 作为输入前缀，并以 $[E]$ 作为输出后缀。2D位置编码表示跨度之间和跨度内部的位置。
 - （d）自注意力掩码。灰色区域被掩盖。Part A中的标记可以参考自己（蓝色框），但不能参考Part B。Part B中的标记可以参考Part A以及它们在B中的前身（黄色和绿色框对应于两个跨度）。
- $[M]$ ：= $[MASK]$ ， $[S]$ ：= $[START]$ ， $[E]$ ：= $[END]$ 。



5.2 模型部署

5.2.1 模型分析

相较于其他开源的LLM大模型，chatGLM-6b针对餐厅评价有如下几点优势：

1. chatglm在预训练过程中加入了大量的中文语料，中文输出效果较好
2. 模型参数量更小（62亿），占用的GPU显存更小和模型推理性能更好。
3. 词表基于中英双语构建，更聚焦。
4. 因为其轻量化的特性，便于本地部署和微调，便于构建embodied AI关于物理世界的感知、理解和交互能力。

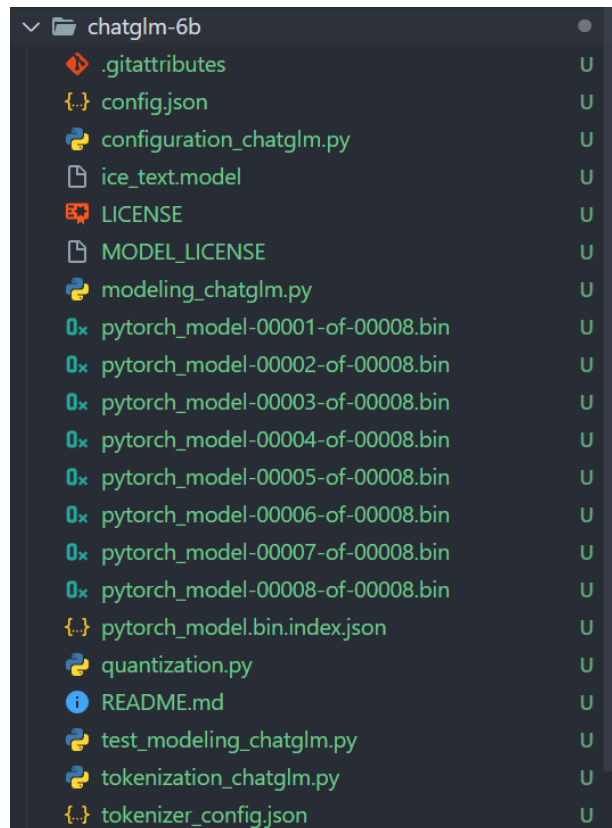
5.2.2 部署操作

1. 首先从Hugging Face Hub 上下载chatglm-6b所需的config等文件

```
1 GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/THUDM/chatglm-6b
```

2. 从<https://cloud.tsinghua.edu.cn/d/fb9f16d6dc8f482596c2/>下载模型参数文件

3. 文件组织形式如下



4. 调用如下代码检查ChatGLM-6B 模型是否能够成功生成推理。

```
1 import os
2 import gradio as gr
3 os.environ["CUDA_VISIBLE_DEVICES"] = "1"
4 from transformers import AutoTokenizer, AutoModel
5
6 class Chatbot():
7     def __init__(self, path):
8         self.path = path
9         self.tokenizer = AutoTokenizer.from_pretrained(path, trust_remote_code=True)
10        self.model = AutoModel.from_pretrained(path, trust_remote_code=True).half()
11        self.model = self.model.eval()
12
13 Chatbot()
```

5.3 ChatGLM-Finetuning

5.3.1 训练数据

训练数据组织成"conversation-response"的格式存放在train.json中，其中，用户输入的conversation为关于餐厅属性评价的text prompt，gpt的response标签为美团ASAP和ai-challenge数据集中所收集的真实用户对于餐厅的评价。

```
1 {
2     "conversation":
3         "生成一段关于餐厅的评价，要包括口味、外观、分量、推荐程度、价格水平、性价比、折扣
        力度、位于商圈附近、交通方便、是否容易寻找、排队时间、服务人员态度、停车方便、点菜/上菜速
        度、装修、嘈杂情况、就餐空间、卫生情况其中的一个或多个方面",
4     "response":
5         "第一次吃马里奥就是在这家店，我记得那时候我还是小学，爸爸妈妈带我去南门口玩，去
        里面挑了几个面包啥的，当时觉得好贵啊！！！！
6 }
```

5.3.2 训练所使用的的依赖

```
1 import os # 用于文件操作
2 from dataclasses import dataclass, field # 用于定义数据类
3 from functools import partial
4 import numpy as np
5 import paddle # 深度学习框架
6 # 自定义的数据处理模块，用于构建数据集
7 from data import (
8     convert_example,
9     custom_instruction_convert_example,
10    read_local_dataset
11 )
12 from sklearn.metrics import accuracy_score # 用于评估准确率
13 from utils import ChatGLMTrainer, save_infer_result # 用于训练模型的一些自定义函数
14 from paddlenlp.data import DataCollatorForSeq2Seq # 用于将输入序列和目标序列对齐并
    进行padding操作。
15 from paddlenlp.datasets import load_dataset # 用于加载数据集
16 from paddlenlp.metrics import BLEU, Rouge1, Rouge2, RougeL # PaddleNLP中的评估指
    标模块，提供了各种评估指标
17 # PaddleNLP中的PEFT模块，用于支持PrefixLM和LoRANet模型的训练和推理。
18 from paddlenlp.peft import LoRAConfig, LoRAModel, PrefixConfig,
    PrefixModelForCausalLM
19 # 提供了各种PrefixLM和LoRANet模型的辅助函数和工具
20 from paddlenlp.peft.prefix import (
```

```

21     chatglm_pad_attention_mask,
22     chatglm_postprocess_past_key_value,
23 )
24 # 训练模块
25 from paddlenlp.trainer import PdArgumentParser, TrainingArguments,
    get_last_checkpoint
26 # transformers模块
27 from paddlenlp.transformers import ChatGLMForConditionalGeneration,
    ChatGLMTokenizer
28 # 记录日志
29 from paddlenlp.utils.log import logger

```

- `data.py` 是自定义的用于数据处理的模块，其中
 - `convert_example` 和 `custom_instruction_convert_example` 函数都是将输入数据转换为模型的输入格式，包括编码、截断、添加特殊token等操作，以便于后续的模型训练和评估。
 - `convert_example` 函数用于将对话样本转换为模型的输入格式，包括对话的问句、答句和历史对话，其输入数据类型为字典，包含"content"或"src"字段和"summary"或"tgt"字段，用于表示对话的问句、答句和历史对话。该函数的目的是为了训练和评估对话生成模型。
 - `custom_instruction_convert_example` 函数用于将自定义指令样本转换为模型的输入格式，其输入数据类型为字典，包含"input"、"instruction"和"output"字段，用于表示自定义指令的输入、指令文本和输出。该函数的目的是为了训练和评估自定义指令生成模型。
 - `read_local_dataset` 函数就是用于读取本地的 `json` 格式的数据集。
- `util.py` 是自定义的一些辅助函数
 - `ChatGLMTrainer` 是一个基于PaddlePaddle框架的对话生成模型训练器，继承自 `PaddlePaddle` 的 `Trainer` 类。该类的作用是封装训练和评估对话生成模型的过程，包括模型推断、生成和评估等功能。
 - 封装了模型的训练过程，包括模型的前向传播、计算损失、反向传播等操作；
 - 封装了模型的评估过程，包括在验证集上计算评估指标和记录日志信息等操作；
 - 封装了模型的预测过程，包括在测试集上生成对话和记录日志信息等操作；
 - 提供了日志函数，用于记录训练过程中的指标和日志信息；
 - 支持自定义参数，包括数据参数、生成参数等；
 - 继承了PaddlePaddle框架中的Trainer类的其他方法和属性，方便用户调用。

- `save_infer_result` 的作用是在对话生成模型训练完成后，使用模型在开发集上进行推断，生成对话，并将生成的对话结果保存在文件中。具体来说，该函数首先从开发集中取出前k个文本样本，然后将这些样本按照每个批次的大小进行划分，然后对于每个批次中的每个文本样本，使用对话生成模型生成对话，并将生成的对话结果保存在一个字典中。最后，将所有字典保存在一个json文件中。该函数的目的是为了通过生成对话的结果来评估对话生成模型的性能。
- 该函数使用了一个名为Predictor的类，该类用于封装使用对话生成模型进行推断的过程，主要实现了以下功能：
 - 封装了对话生成模型的推断过程，包括对输入进行编码、解码和生成等操作；
 - 支持对单个文本或多个文本进行推断；
 - 支持设置输入文本的长度限制；
 - 提供了输出结果的格式化方法，可以将生成的对话结果转换为易读的文本形式。

5.3.3 训练过程

训练ChatGLM模型的过程，主要包括以下几个步骤：

1. 定义模型参数、数据参数和训练参数的配置。其中模型参数包括模型名称和LoRA、Prefix等技术的使用与否；数据参数包括数据集路径、源文本和目标文本的最大长度、beam search的beam大小等；训练参数包括训练设备类型、学习率、batch size等。
2. 加载预训练的ChatGLM模型，并根据模型参数配置是否使用LoRA或Prefix等技术来进行模型的修改和微调。
3. 加载数据集，并根据数据参数配置进行转换和处理。
4. 定义DataCollator，用于将数据转换成模型的输入格式。同时定义 `compute_metrics` 函数，用于评估模型效果。
5. 定义Trainer，并根据训练参数配置进行训练和评估。训练过程中可以使用自动混合精度（Automatic Mixed Precision, AMP）技术加速训练。训练完成后，保存训练好的模型、评估结果等。如果需要，还可以生成一些示例文本进行演示。

5.4 模型的使用

- 在 `gpt.py` 中封装了一个 `Chatbot()` 类，使用模型路径初始化，其中包括 `tokenizer`，并将模型调至评估模式。
- 在 `chat` 标签页中，输入框允许用户输入对话内容或问题，并且点击 `Generate` 按钮后，聊天机器人将会生成回答并在输出框中显示。同时，用户也可以在 `superAgent` 子标签页中输入一段关于餐厅的评论，并点击 `Give Suggestions` 按钮，聊天机器人将为用户提供对餐厅改进的建议。

- 在代码中，`botactivation()` 函数用于启动聊天机器人，`chatbot()` 函数用于生成聊天机器人的回答并将对话记录写入日志文件中。在 `Super_agent()` 函数中，用户可以输入餐厅评论，并将其与预设的提示语一同输入，聊天机器人将为用户提供改进建议。聊天记录会写入到 `chat_logs/logs.txt` 文件中，并且纯文本记录会写入到 `chat_logs/pure_logs.txt` 文件中，以便后续的模式训练。
- 用户可以点击 `start robot!` 按钮来启动聊天机器人，然后在 `chat` 标签页中开始对话。

六、基于gradio的交互式界面

运行系统在 `./unified_sentiment_extraction/liushuai_workdir/` 目录中使用 `python3 app.py` 运行界面。运行这段代码（`app.py`）后，会启动一个基于 Gradio 的 Web 应用程序。这个应用程序包含三个标签页：`chat`、`history` 和 `rating system`。

6.1 用户评价及交互界面

- 在 `chat` 标签页中，用户可以与聊天机器人进行交互。在文本框中输入问题或对话内容，然后点击 `Generate` 按钮，聊天机器人就会生成回答，并将回答显示在输出框中。用户也可以点击 `Clear` 按钮清空输入框中的文本。在 `superAgent` 子标签页中，用户可以输入一段关于餐厅的评论，然后点击 `Give Suggestions` 按钮，聊天机器人会给出对餐厅改进的建议。
- 在 `history` 标签页中，用户可以查看之前的聊天记录。点击 `View history` 按钮，聊天记录会显示在输出框中。

chat

history

rating system

Textbox

作为餐厅老板，我会根据用户的评论反馈，对餐厅进行全面的改进。以下是我可能会采取的一些改进措施：

1. 环境改进：我将考虑对餐厅的环境进行改进，例如增加一些优雅舒适的座位，或者引进一些新的装饰元素，提升餐厅的整体氛围。

2. 菜品创新：我将鼓励厨师团队进行菜品创新，尝试推出一些更加特色、创新的菜品，吸引更多食客的关注。

3. 服务提升：我将加强员工的培训，提升他们的服务技能和素质，提供更好的服务体验，让食客感到更加舒适和满意。

4. 加强品质控制：我将加强对菜品的采购和加工控制，确保所有的菜品都符合标准，口感和质量都能够得到保障。

5. 加强营销宣传：我将加大营销宣传的力度，通过各种渠道和方式吸引食客的注意，提高餐厅的知名度和美誉度。

综上所述，我会采取多种措施，不断提升餐厅的品质和服务体验，以满足食客的需求和期望，让更多的人喜欢和选择我们的餐厅。

Textbox

生成一段关于餐厅的评价，要包括口味、外观、分量、推荐程度、价格水平、性价比、折扣力度、位于商圈附近、交通方便、是否容易寻找、排队时间、服务人员态度、停车方便、点菜/上菜速度、装修、嘈杂情况、就餐空间、卫生情况其中的一个或多个方面

生成评价

清除

superAgent

位置在宇宙中心五道口地铁站附近。位于五道口国际美食苑的2楼。和其它分店不一样，一进门特别大一个厅，感觉像个大食堂，环境有点吵闹。饭点是一定要等位的，好在叫号非常快。分析第一是因为日昌本身人气就旺，第二因为他家菜量普遍很大，性价比还比较高。
【豉汁排骨饭】推荐，排骨很多很入味。汤汁不够浓还可以要。服务员会给你拌好，最爱吃底下的锅巴，很香。两个人吃一份就够了。尝试过其他几种煲仔饭还是觉得排骨的最好吃。
【腊味煲仔饭】很大一碗，腊肉很多，就是有点油。
【蟹黄豆腐】蟹粉吃起来沙沙的。味道浓郁，豆腐很嫩。放在纸炉上加热，味道不错，推荐。
【杨枝甘露】本身就很好，别家都是一小碗，这的杨枝甘露是成扎的，喝起来很过瘾，清凉爽口！非常好喝！每次都眼大肚子小的点个大扎，最后喝不了.....
【纸包鸡翅】必点菜品，鸡翅烤出来焦香焦香的，肉很入味很嫩，因为被锡纸包住烤的，吃的时候小心被烫到。味道有点甜又有点辣，很好吃。两个人点一份合适，吃多了会有点腻。
总体来说是个很适合聚餐的地方，饭菜味道普遍都还不错。但去过几次发现，水平略有下降，这点希望改进。还有作为粤菜馆他家的粥非常一般.....重点提醒：日昌有很多假冒的，吃过一次，味道差太多，一定要认准。

生成反馈

Give Suggestions

清除

start robot!

Textbox

success!

启动chatbot成功

chat

history

rating system

Textbox

You: 生成一段关于餐厅的评价，要包括口味、外观、分量、推荐程度、价格水平、性价比、折扣力度、位于商圈附近、交通方便、是否容易寻找、排队时间、服务人员态度、停车方便、点菜/上菜速度、装修、嘈杂情况、就餐空间、卫生情况其中的一个或多个方面

Bot: 这家餐厅位于市中心，靠近商场和交通枢纽，提供了多种口味的美食。餐厅的外观简洁大方，装修现代化，让人感到舒适和放松。餐厅里的食物味道很好，分量也足够，价格也很实惠，性价比非常高。我推荐这家餐厅给朋友，他们也喜欢这里的美食。除了美食，餐厅的交通也非常便利，靠近商场和交通枢纽，很容易找到。餐厅还提供停车位，非常方便。服务员的态度也非常好，非常热情友好，让人感到非常舒适。不过，餐厅里有时候会出现一些嘈杂的情况，可能会影响到就餐体验。餐厅的就餐空间比较小，有些拥挤，但总的来说还是非常方便和舒适的。卫生情况也比较好，餐厅的环境也比较干净，让人感到很放心。总的来说，这家餐厅在口味、外观、分量、推荐程度、价格水平、性价比、折扣力度、位于商圈附近、交通方便、是否容易寻找、服务人员态度、停车方便、点菜/上菜速度、装修、嘈杂情况、就餐空间、卫生情况等方面都非常优秀。如果有机会去这家餐厅用餐，我会强烈推荐。

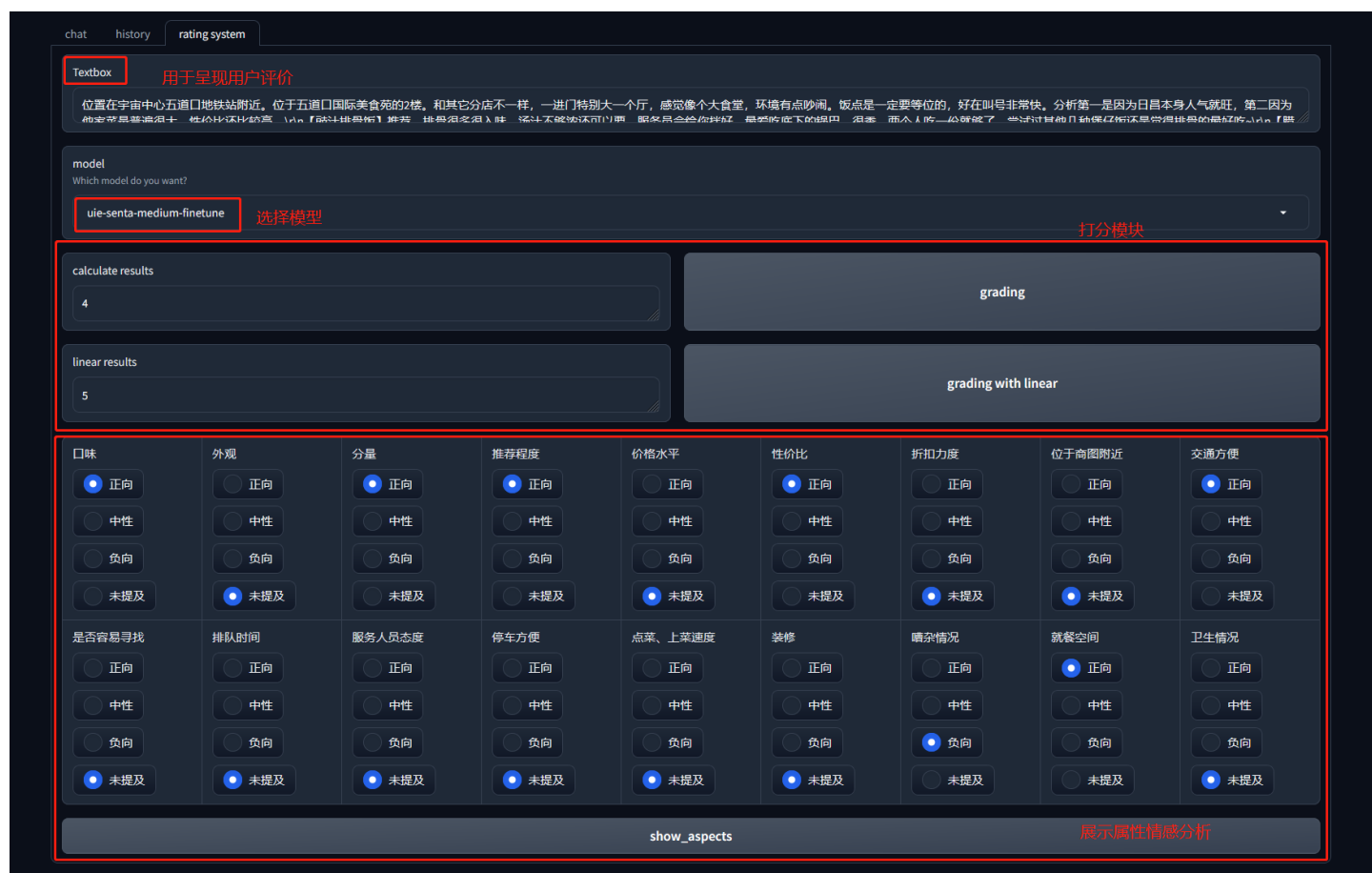
You: 生成一段关于餐厅的评价，要包括口味、外观、分量、推荐程度、价格水平、性价比、折扣力度、位于商圈附近、交通方便、是否容易寻找、排队时间、服务人员态度、停车方便、点菜/上菜速度、装修、嘈杂情况、就餐空间、卫生情况其中的一个或多个方面

Bot: 评价餐厅需要综合考虑多个方面，以下是一份可能的评价：这家餐厅位于商圈附近，装修简洁大方，环境舒适宜人。餐厅口味非常好，菜肴口感丰富，味道鲜美，让人回味无穷。分量也足够大，让人感觉非常满足。推荐程度很高，值得一试。价格水平也比较适中，不算很高，但也不算太低。性价比非常高，非常划算。折扣力度也比较合适，经常推出优惠活动，让人倍感实惠。位于商圈附近，交通便利，很容易找到，交通也非常便利，距离地铁和公交站都很方便。排队时间较短，服务态度也很好，让人感觉非常舒适。餐厅的点菜/上菜速度也非常快，让人感觉非常方便快捷。餐厅的装修也很舒适，氛围很好，让人感到很放松。就餐空间也比较大，让人感觉非常舒适。餐厅的卫生情况也非常好，环境干净整洁，让人感觉非常舒适。总的来说，这家餐厅口味好，价格合适，交通便利，服务态度好，就餐空间大，卫生情况好，是一个不错的选择。

View history

6.2 打分系统

- 在 rating system 标签页中，用户可以对餐厅评论进行情感评价。在文本框中输入评论内容，然后选择一个模型（例如 uie-senta-medium-finetune），点击 grading 按钮，评分结果会显示在输出框中。评分结果包括一个整数（表示总体评价），以及若干个单选框（用于选择评论中不同方面的评价）。在 show_aspects 按钮下方会显示各个方面的评价。



七、实验反思与总结

7.1 未来可以优化的方向

7.1.1 实验中发现的不足之处

1. 在运行过程中有可能会遇到系统错误，这是因为我们没有考虑到paddle和transformer在环境中的不可共存性。当环境中同时安装了 `paddle` 和 `torch` 等深度学习框架时会出现 `OSError`。但是经过实验，我们的解决方案是先进行评分（使用 `paddle`、`paddlenlp`）后，在 `chat` 页面进行评价（使用 `transformer`）。这样可以避免环境冲突。
2. 在实验过程中出现了一个小插曲：在验收前几天编写代码的环境突然出错，导致我们的文件都没有了。然后在代码重现和模型训练过程中花费了很多时间。因此模型还没达到原先我们训练出的最佳状态，而且在评分模型上也有所欠缺。以后在编程过程中应该养成及时备份的习惯。
3. 使用的数据中 `positive` 的比重过高，预测出的结果还可以优化应当对数据进行适当增广。

7.1.2 系统的优点

1. 借助 `chatGPT` 模型一键生成评价，可以让用户仅提供关键词，减少编写评论的耗时。在店家需要评价、用户赶时间的情况下可以节省精力。
2. 通过 `chatGPT` 自动生成店家回复，可以解决店家不能及时回复的问题，并且在数据充足的情况下可以给用户比较合理的反馈。店家只需要对回答进行修订，这样既可以减少人力资源消耗又可以更加全面地给店家和用户反馈。

3. 一键打分采用统一的标准，这样可以避免用户自行评分对评分效果的不确定性影响（例如有些用户认为五星是好评，三星是一般；但是有些用户认为四星才是一般，三星是较差），减少了在数据统计中带来的麻烦。

7.1.3 未来可能应用的场景

我们的系统可以应用到餐饮行业，同样也可扩展到其余类似的评论行业（影视、娱乐、电商等）。

1. 可以应用到美团、大众点评等平台上，并给予统计数据给用户推荐评分较高的店家。
2. 将用户评价数据提取特征向量，并利用系统过滤等算法给用户推荐合适的店家。
3. 可以根据用户给出的历史评价生成用户画像，并总结出用户喜欢与不喜欢的店家类型。

在Embodied AI的技术方面，我们的系统可以做以下优化。

1. 感知能力的增强：将视觉和语音识别技术嵌入到系统中，使系统能够感知和理解用户在餐厅内的行为和语言，以更加智能、自然地与用户进行交互。
2. 自主决策和行动能力的增强：将强化学习、深度强化学习等技术嵌入到系统中，使系统能够自主决策和执行任务，如自动点菜、送餐、结账等，从而提高服务效率和用户体验。
3. 人机协同能力的增强：将协同机器人、人机协同等技术嵌入到系统中，使系统能够与人类进行协同工作，如清洁、排桌、调餐等，从而提高工作效率和质量。
4. 安全性和隐私保护的加强：在系统中加入安全和隐私保护机制，确保用户数据和隐私的安全，避免数据泄露、滥用等问题。
5. 可持续性和可扩展性的提高：采用可持续和可扩展的技术架构，确保系统能够持续稳定地运行，并能够适应未来的技术发展和业务需求。

7.2 实验总结

在系统功能上，餐饮评价系统是一个非常实用的系统，能够对用户的评价进行全面、细致的分析，并且能够根据分析结果生成店家对用户的回复。在这个过程中，我们使用了多种技术和工具，包括 ASAP 数据、UIE 模型、线性回归、简单平均、chatGLM-6B 等。

- 在实验中，我们首先使用 ASAP 数据进行了训练，使用 UIE 模型对用户评论进行了属性级情感分析，并根据线性回归、简单平均等方式生成了最终评分。这一步的结果比较准确，能够对用户评论进行全面、准确的分析，并且能够生成较为合理的评分。
- 接下来，我们使用 chatGLM-6B 模型进行自动生成评论和店家回复。在这个过程中，我们遇到了一些问题，比如模型训练时间较长、对话生成的结果不够准确等。不过，通过不断的优化和调整，我们最终还是得到了比较满意的结果。

总的来说，这个实验让我们对于自然语言处理技术有了更深入的了解，并且让我们学会了如何使用这些技术和工具来实现一个实用的系统。在未来的工作中，我们可以继续优化模型，提高系统的准确性和实用性。

八、参考文献及链接

8.1 数据来源（考察过的网址）

8.1.1 国内

<https://github.com/Meituan-Dianping/asap>

(*NAACL 2021 paper: ASAP: A Chinese Review Dataset Towards Aspect Category Sentiment Analysis and Rating*)

ASAP 是一个大规模的中文餐厅评论数据集，用于面向方面类别情感分析（ACSA）和评论评分预测（RP），包括来自大众点评 App 的 46,730 条真实用户评论。除了一个 5 星级评分，每条评论还根据其 18 个预定义的方面类别（例如食物、服务、环境等）的情感极性进行手动注释。我们将数据集随机分成训练集（36,850 条）、验证集（4,940 条）和测试集（4,940 条）。

<https://github.com/YZHANG1270/Aspect-Based-Sentiment-Analysis/tree/master> Aspect Based Sentiment Analysis 基于方面的细粒度情感分析

8.1.2 国外

[Amazon.com](https://www.amazon.com)

<https://www.trustpilot.com/>

8.2 模型相关

UIE论文 <https://arxiv.org/abs/2203.12277>

知识蒸馏论文 [Distilling the Knowledge in a Neural Network](#)

chatGLM-6B参考仓库地址 <https://github.com/THUDM/ChatGLM-6B>

8.3 Prompt engineering

<http://t.csdn.cn/k8B3l>

<http://t.csdn.cn/gMMAQ>

<http://t.csdn.cn/fAK9l>