

知识表示与推理课程

命名实体识别实验报告

姓名：张智雄

学号：2021112845

日期：2024.5.22

一、基于 ltp 的命名实体识别

(1. 简述命名实体识别相关概念。)

命名实体识别 (Named Entity Recognition, NER) 是自然语言处理 (Natural Language Processing, NLP) 中的一个基本任务, 旨在从文本中识别和分类命名实体。命名实体一般指的是文本中具有特定意义或者指代性强的实体, 通常包括人名、地名、组织名、日期、时间、货币金额、百分比等。

命名实体识别任务通常包括两部分:

- 实体的边界识别
- 确定实体的类型 (人名、地名、机构名或其他)

NER 系统就是从非结构化的输入文本中抽取出上述实体, 并且可以按照业务需求识别出更多类别的实体。主要包含以下概念

- 命名实体: 指在特定上下文中具有明确意义的实体, 如人名 (John Doe)、地名 (New York)、组织名 (UN)、日期 (2023 年 5 月 22 日)、货币金额 (\$100) 等。
- 标注 (Annotation): 将文本中的命名实体用特定的标签标记出来。例如, 在句子 “John Doe lives in New York.” 中, John Doe 被标记为人名 (PERSON), New York 被标记为地名 (LOCATION)。
- 识别 (Recognition): 自动识别并标注文本中的命名实体。这通常通过机器学习或规则匹配的方法实现。
- 分类 (Classification): 将识别出的实体归类到预定义的类别中, 如人名、地名、组织名等。

同时, NER 具有较为广泛的引用场景:

- 信息抽取: 从大量文本中提取关键信息, 如新闻摘要、法律文档分析等。
- 问答系统: 提高问答系统的理解能力, 如在问题中识别提问的具体实体。
- 文本分类: 增强文本分类系统对实体的理解, 如在社交媒体上分类不同主题的帖子。
- 情感分析: 结合实体识别, 可以更准确地分析特定实体的情感倾向。

(2. 更改例句, 使用 ltp 库完成命名实体识别, 分析对应识别结果。)

例句 1: “小明同学于今年暑假游览武汉。”

- 小明: 人名
- 武汉: 地名

```
Loading weights from local directory  
[[('Nh', '小明'), ('Ns', '武汉')]]
```

例句 2: “马云于 1964 年 9 月 10 日出生在浙江省杭州市。”

- 马云: 人名
- 浙江省: 地名
- 杭州市: 地名

```
Loading weights from local directory  
[[('Nh', '马云'), ('Ns', '浙江省'), ('Ns', '杭州市')]]
```

例句 3：“腾讯公司总部位于中国深圳。”

- 腾讯公司：组织名
- 中国：组织名
- 深圳：地名

```
Loading weights from local directory
[[('Ni', '腾讯公司'), ('Ni', '中国'), ('Ns', '深圳')]]
```

例句 4：“刘挺老师出席了在北京举行的国际会议。”

- 刘挺：人名
- 北京：地名

```
Loading weights from local directory
[[('Nh', '刘挺'), ('Ns', '北京')]]
```

二、基于 bert 的命名实体识别

- (1. 补全各部分代码，完成手册中三部分实验内容，在报告中以截图方式展示补全代码。)

Process.py

首先定义相关文件初始地址如下：

```
15     ## todo 定义相关文件初始地址
16     self.data_path = "./data/duie_data/"
17     self.train_file = self.data_path + "train.json"
18     self.dev_file = self.data_path + "dev.json"
19     self.test_file = self.data_path + "test.json"
20     self.schema_file = self.data_path + "duie_schema.json"
```

而后根据数据格式按字典读取相关键值信息：

```
29     ## todo 基于模版文件 获取subject_type
30     subject_type = data["subject_type"]
31     ## todo 基于模版文件 获取object_type
32     object_type = data["object_type"]["@value"]
33     if "人物" in subject_type:
34         subject_type = "人物"
35     if "人物" in object_type:
36         object_type = "人物"
```

而后根据提取出的三元组关系（subject-predicate-object, SPO）在文本 text 中查找特定的“主体”（subject）和“对象”（object）的位置，并根据它们的类型给文本中的这些位置打上标签。标签的格式为 B-`{subject_type}`（对于开始的字符）和 I-`{subject_type}`（对于中间的字符）。

➤ **查找主体（subject）：**首先使用正则表达式 `re.finditer` 在 text 中查找 `spo['subject']` 的值。为了避免正则表达式中的特殊字符问题，使用 `re.escape` 对 `spo['subject']` 进行转义。如果 `spo['subject_type']` 中包含“人物”，则将 `subject_type` 设置为“人物”。

➤ **查找对象（object）：**同样使用正则表达式查找 `spo['object']['@value']` 的值，如果 `spo['object_type']['@value']` 中包含“人物”，则将 `object_type` 设置为“人物”。

对于找到的每个主体匹配项，会在文本中相应位置打上标签。而如果在查找过程中发生异常（如正则表达式错误、spo 字典结构问题等），则捕获异常并打印出相关信息，然后退出程序。

```

67         try:
68             # todo 找到subject在文本中的位置
69             subject_re_res = list(re.finditer(re.escape(spo['subject']), text))
70             subject_type = spo["subject_type"]
71             if "人物" in subject_type:
72                 subject_type = "人物"
73         except Exception as e:
74             print(e)
75             print(spo['subject'].replace('+', '\+'), line['text'])
76             import sys
77             sys.exit(0)
78         for sbj in subject_re_res:
79             sbj_span = sbj.span()
80             sbj_start = sbj_span[0]
81             sbj_end = sbj_span[1]
82             tmp["labels"][sbj_start] = f"B-{subject_type}"
83             for j in range(sbj_start + 1, sbj_end):
84                 tmp["labels"][j] = f"I-{subject_type}"
85         try:
86             # todo 找到object在文本中的位置
87             object_re_res = list(re.finditer(re.escape(spo['object']['@value']), text))
88             object_type = spo['object_type']['@value']
89             if "人物" in object_type:
90                 object_type = "人物"
91         except Exception as e:
92             print(e)
93             print(line)
94             print(spo['object']['@value'].replace('+', '\+').replace('(', '('), line['text'])
95             import sys
96             sys.exit(0)

```

最后在 main 函数中补全实际的保存地址为:

```

114     ## todo 补全训练集和验证集的处理结果保存地址
115     processDuiData.get_ner_data(processDuiData.train_file,
116                                os.path.join(processDuiData.data_path, 'ner_data/train.txt'))
117     processDuiData.get_ner_data(processDuiData.dev_file,
118                                os.path.join(processDuiData.data_path, 'ner_data/dev.txt'))

```

Model.py

这里主要涉及到两个改动:

- 从 input_ids 的维度中获取批量大小 batch_size。
- 使用一个线性层 (self.linear) 对 BiLSTM 的输出进行变换。这个线性层可能用于将 LSTM 的输出映射到与标签集大小相匹配的维度。

```

29     ## todo 获取批量大小
30     batch_size = input_ids.size(0)
31     seq_out, _ = self.bilstm(seq_out)
32     seq_out = seq_out.contiguous().view(-1, self.lstm_hidden * 2)
33     seq_out = seq_out.contiguous().view(batch_size, self.max_seq_len, -1)
34     ## todo 使用线性层进行最后的预测
35     seq_out = self.linear(seq_out)

```

Data_loader.py

这里主要涉及到两个填充:

- 对于 input_ids, 代码首先确定当前 tmp_input_ids 列表的长度是否小于预设的最大序列长度 self.max_seq_len。如果小于, 就在列表的末尾添加指定数量的 PAD Token 对应的 id 来填充。
- 而对于对于 attention_mask, 如果其长度也小于 self.max_seq_len, 则需要用 0 将其补全至与 input_ids 相同的长度。0 表示不参与注意力计算。

```

26     ## todo 补全input_ids至最大长度 0 是 pad_token
27     input_ids = tmp_input_ids + (self.tokenizer.convert_tokens_to_ids(["PAD"]))
28     * (self.max_seq_len - len(tmp_input_ids))
29     ## todo 补全attention mask至最大长度
30     attention_mask = attention_mask + [0] * (self.max_seq_len - len(attention_mask))

```

Main.py

加入梯度传播和更新的相关代码，并按刚才 Process.py 构建数据集的类和 DataLoader 加载数据集。

```

55     # todo 梯度清零
56     self.optimizer.zero_grad()
57     # todo 反向传播计算梯度
58     loss.backward()
59     # todo 更新参数
60     self.optimizer.step()
61     # todo 更新学习率
62     self.scheduler.step()
63     print(f"【train】 {epoch}/{self.epochs} {global_step}/{self.total_step} loss:{loss.item()}")

```

```

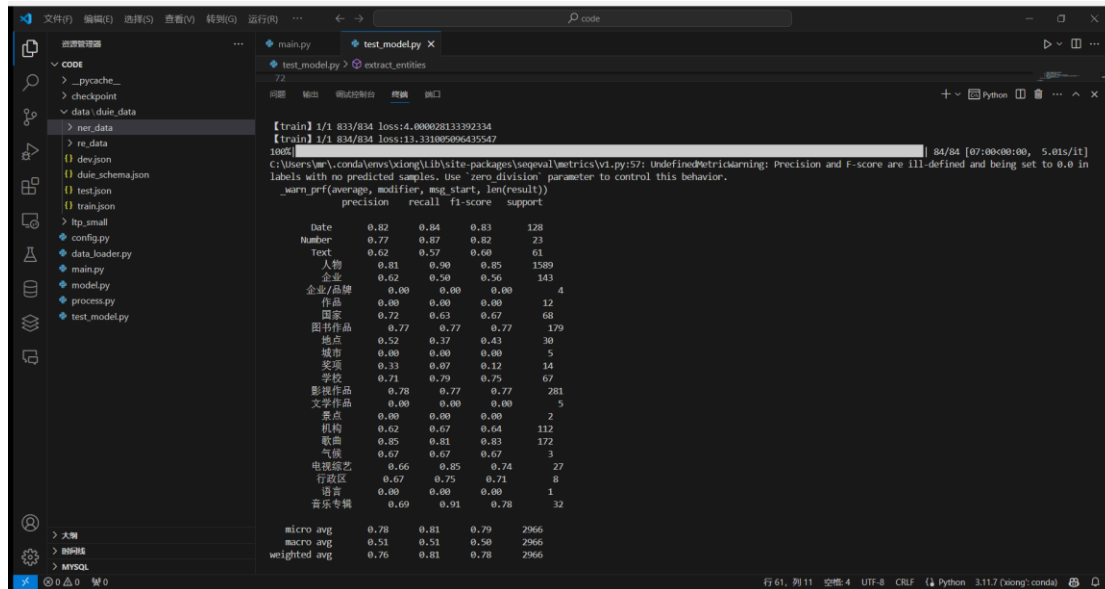
160     # todo 基于NerDataset 加载数据集
161     train_dataset = NerDataset(train_data, args, tokenizer)
162     dev_dataset = NerDataset(dev_data, args, tokenizer)
163     train_loader = DataLoader(train_dataset, shuffle=True, batch_size=args.train_batch_size, num_workers=2)
164     dev_loader = DataLoader(dev_dataset, shuffle=False, batch_size=args.dev_batch_size, num_workers=2)

```

(2. 对于训练部分给出训练成功截图，并分析模型训练代码逻辑。)

这段代码的训练逻辑主要是通过 `Trainer` 类来执行模型的训练和评估过程，其中使用了 PyTorch 框架提供的 DataLoader、Optimizer 和 Scheduler 等功能来实现数据加载、优化和学习率调整。

1. 根据数据集名称参数 data_name 使用 NerConfig 类初始化 args 对象，并将参数配置保存到 ner_args.json 文件中。
2. 使用 BertTokenizer 从预训练的 BERT 模型目录中加载分词器，并导入到指定的 device (GPU 或 CPU)。
3. 从数据集目录中读取训练集和验证集的数据，并将其转换为 JSON 格式。
4. 使用 NerDataset 类加载训练集和验证集的数据，该类将数据转换为模型可以处理的格式，并进行分批处理。最后通过 DataLoader 对训练集和验证集的数据进行迭代加载。
5. 创建 BERT-based 的命名实体识别模型 (BertNer 类的实例)，将模型移到相应的 device 上 (GPU 或 CPU)。
6. 计算总的训练步数，并根据这个步数构建优化器和学习率调度器。
7. 使用 Trainer 类进行训练 (train.train() 方法)。Trainer 类是一个自定义的训练器，接收模型、数据加载器、优化器、调度器等参数，用于实际的模型训练过程。
8. 完成训练后，调用 train.test() 方法对模型进行评估，获取评估结果。



```
【train】 1/1 833/834 loss:4.00002813392334
【train】 1/1 834/834 loss:13.331005096435547
100%
C:\Users\m\conda\envs\yolov11\site-packages\torch\nn\modules\losses.py:57: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use 'zero_division' parameter to control this behavior.
warn_pre(average, modifier, msg_start, len(result))
precision    recall  f1-score   support

   Date       0.82    0.84    0.83    128
   Number     0.77    0.87    0.82     23
   Text       0.62    0.57    0.60     61
   人物       0.81    0.90    0.85    1589
   企业       0.62    0.50    0.56    143
   企业/品牌   0.00    0.00    0.00     4
   作品       0.00    0.00    0.00    12
   国家       0.72    0.63    0.67     68
   图书作品    0.77    0.77    0.77    179
   地点       0.52    0.37    0.43     30
   城市       0.00    0.00    0.00     5
   奖项       0.33    0.07    0.12    14
   学校       0.71    0.79    0.75     67
   影视作品    0.78    0.77    0.77    281
   文学作品    0.00    0.00    0.00     5
   景点       0.00    0.00    0.00     2
   机构       0.62    0.67    0.64    112
   歌曲       0.85    0.81    0.83    172
   气象       0.67    0.67    0.67     3
   电视综艺    0.66    0.85    0.74     27
   行政区     0.67    0.75    0.71     8
   语言       0.00    0.00    0.00     1
   音乐专辑    0.69    0.91    0.78     32

 micro avg     0.78    0.81    0.79    2966
 macro avg     0.51    0.51    0.50    2966
weighted avg     0.76    0.81    0.78    2966
```

（3. 提供预测部分代码，并给出手册中例句的命名实体识别结果。）

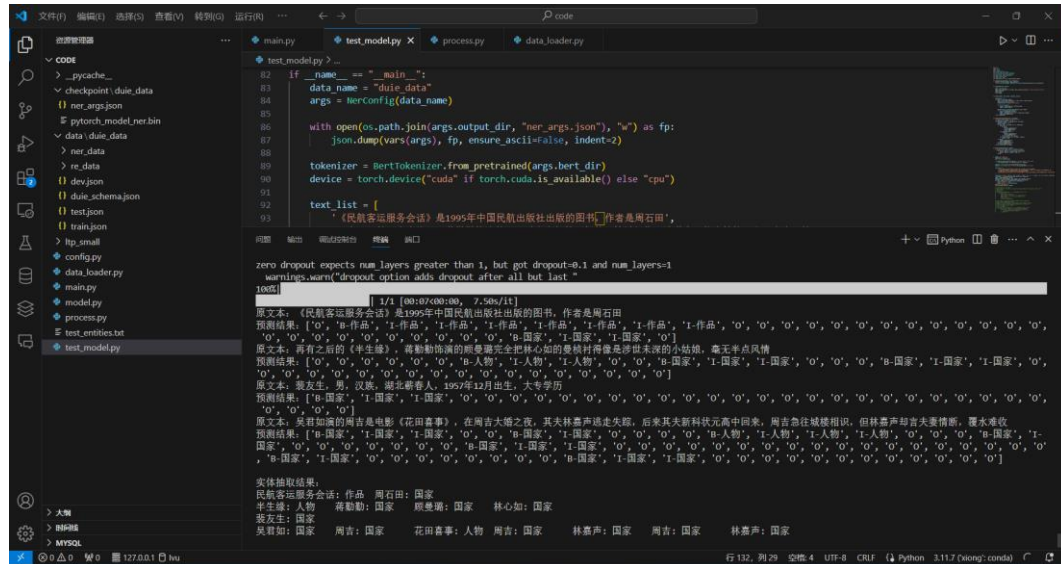
预测部分代码逻辑如下：

1. 首先，通过加载模型所需的参数配置和预训练的 BERT 分词器，准备好待预测的文本数据（按字符划分，label 全初始化构造为 O）。使用 NerDataset 和类将数据构造为输入的测试集。
2. 而后，初始化 BertNer 模型实例，加载模型的权重(pytorch_model_ner.bin)，将模型移动到指定设备(GPU 或 CPU)，并将模型设置为评估模式(model.eval())。
3. 评估时，获取测试集的输入张量(input_ids)和注意力掩码(attention_mask)在不计算梯度的上下文中（with torch.no_grad()）运行模型，得到输出的 logits。根据注意力掩码的长度，提取每个样本的预测标签，将 ID 转换成对应的标签。
4. 最后，逐字符遍历每个预测标签和原始数据，识别并提取实体及其标签。当遇到标签为'O'时，结束当前实体提取；遇到'B-'开头的标签时，开始新实体；遇到'I-'开头的标签且与当前实体标签一致时，继续当前实体。

```
82 if __name__ == "__main__":
83     data_name = "duie_data"
84     args = NerConfig(data_name)
85
86     with open(os.path.join(args.output_dir, "ner_args.json"), "w") as fp:
87         json.dump(vars(args), fp, ensure_ascii=False, indent=2)
88
89     tokenizer = BertTokenizer.from_pretrained(args.bert_dir)
90     device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
91
92     text_list = [
93         '《民航客运服务会话》是1995年中国民航出版社出版的图书，作者是周石田',
94         '再有之后的《半生缘》，蒋勤勤饰演的顾曼璐完全把林心如的曼桢衬得像是涉世未深的小姑娘，毫无半点风情',
95         '裴友生，男，汉族，湖北蕲春人，1957年12月出生，大专学历，
96         '吴君如演的周吉是电影《花田喜事》，在周吉大婚之夜，其夫林嘉声逃走失踪，后来其夫新科状元高中回来，周吉急忙进城相迎，但林嘉声却言夫妻情断，覆水难收'
97     ]
98
99     test_data = [{'text': [char for char in text], 'labels': ['O' for _ in text], 'id': 0} for text in text_list]
100     test_loader = create_data_loader(test_data, args, tokenizer)
101     model = load_model(args, device)
102     predictions = predict(model, test_loader, args.id2label, device)
103     entities, labels = extract_entities(predictions, test_data)
104     for text, prediction in zip(text_list, predictions):
105         print("原文本: " + text)
106         print("预测结果: " + str(prediction))
107     print("\n\n实体抽取结果")
108     print_entities(entities, labels)
```


对于指导手册给的用例，最后预测得到的结果和截图如下：

1. 民航客运服务会话: 作品 周石田: 国家
2. 半生缘: 人物 蒋勤勤: 国家 顾曼璐: 国家 林心如: 国家
3. 裴友生: 国家
4. 吴君如: 国家 花田喜事: 人物 周吉: 国家 林嘉声: 国家



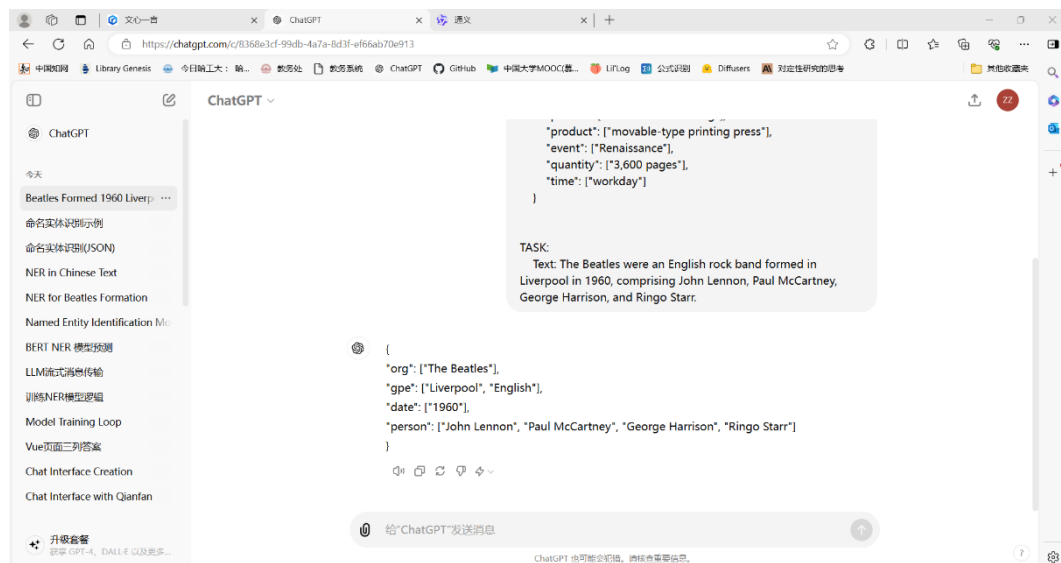
三、 基于大模型的命名实体识别

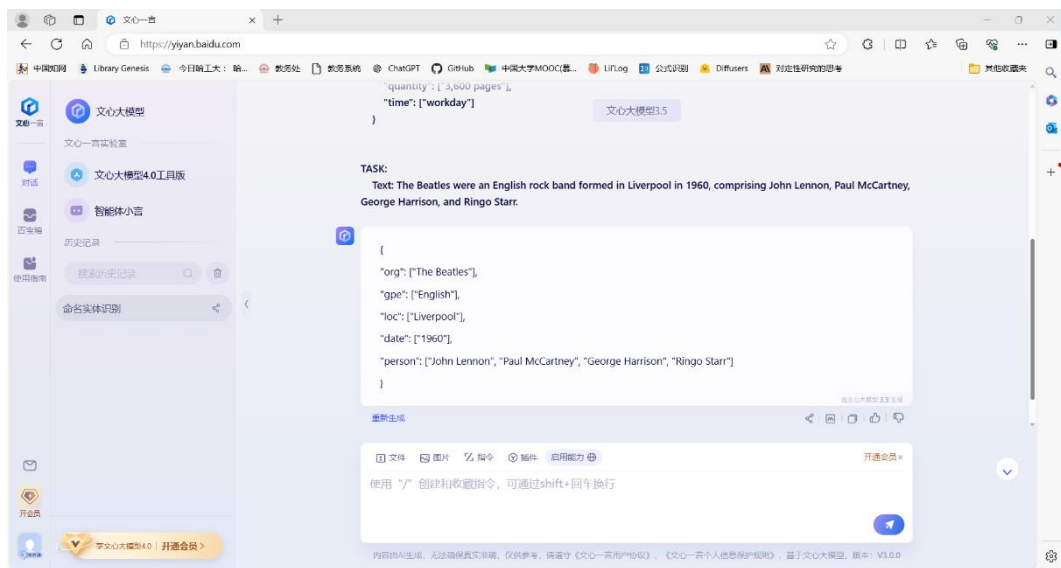
- (1. 至少采用两种大模型完成例句的命名实体识别, 并按照 json 格式展示实验结果)

本实验采用两种格式类似的中英文双语 One-shot prompt 和 Zero-shot prompt (详情见附录和附件) 进行基于大模型的命名实体识别, 结果如下:

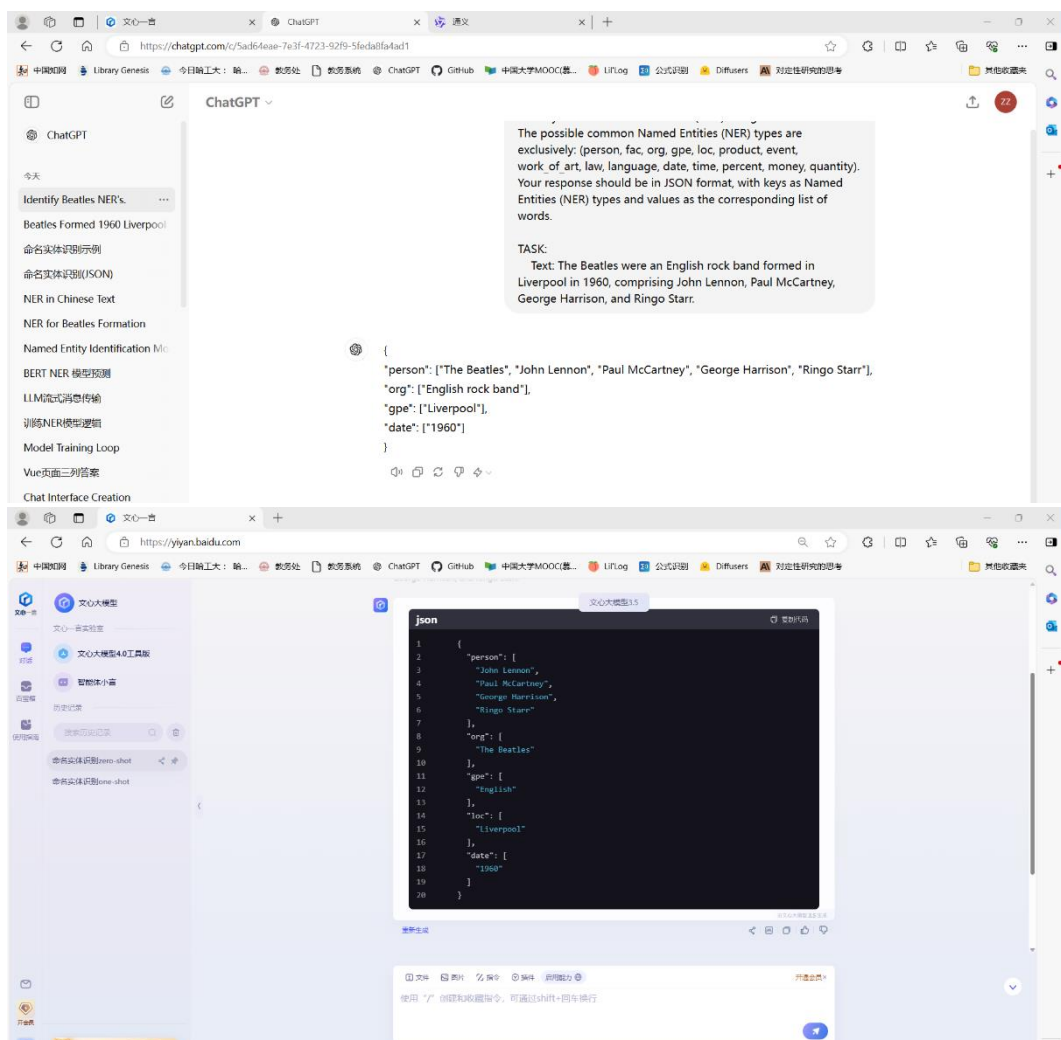
例句 1: *The Beatles were an English rock band formed in Liverpool in 1960, comprising John Lennon, Paul McCartney, George Harrison, and Ringo Starr.*

使用 GPT-3.5，文心大模型 3.5 的 **One-shot** 获得结果如下：





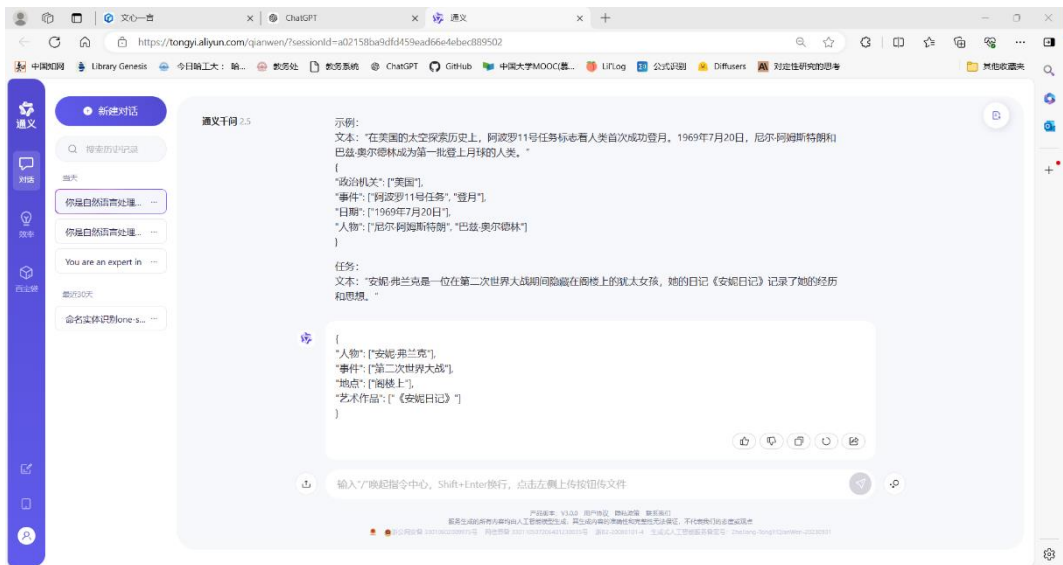
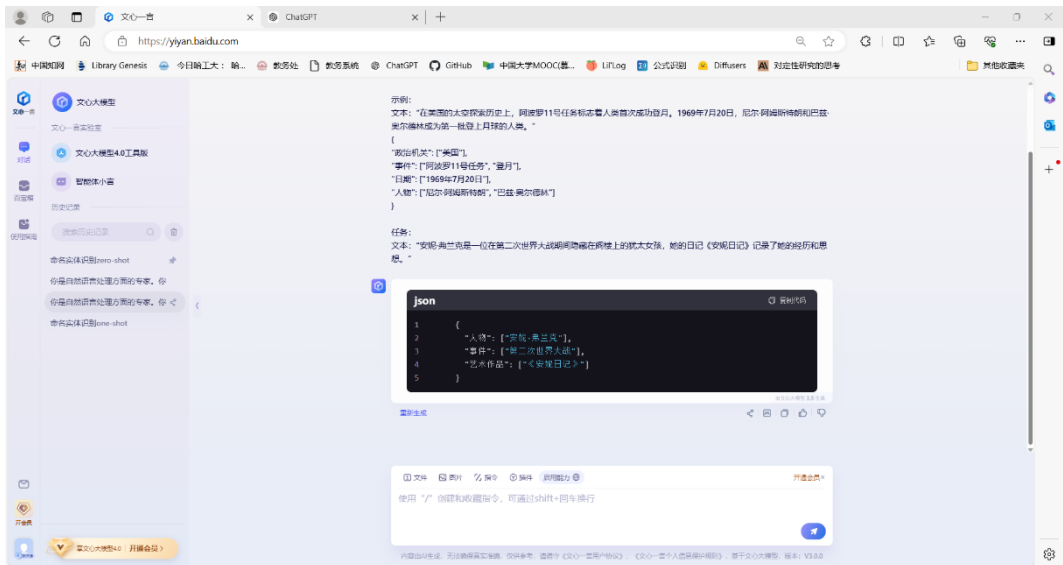
使用 GPT-3.5，文心大模型 3.5 的 **Zero-shot** 获得结果分别如下：



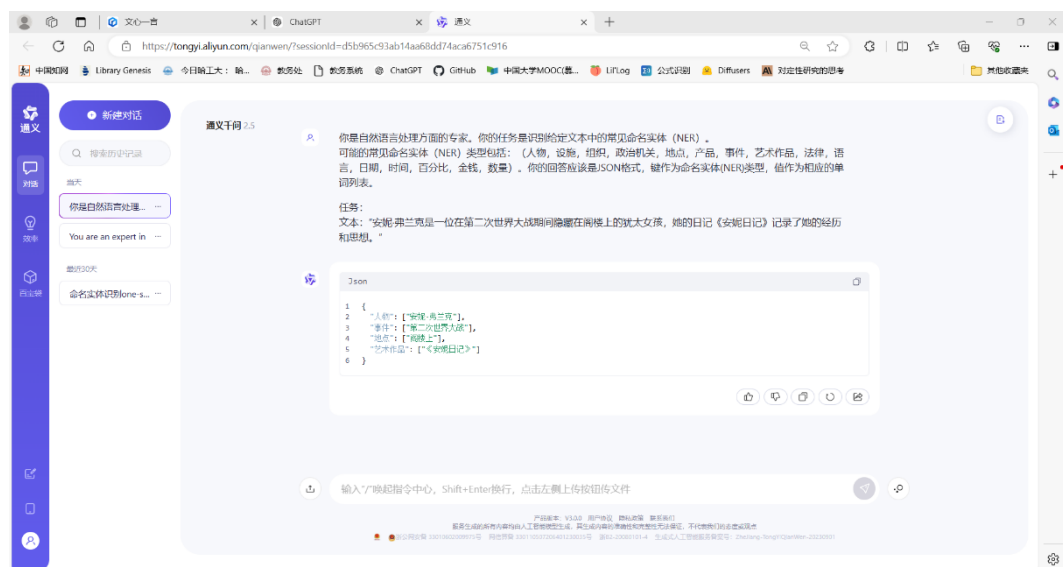
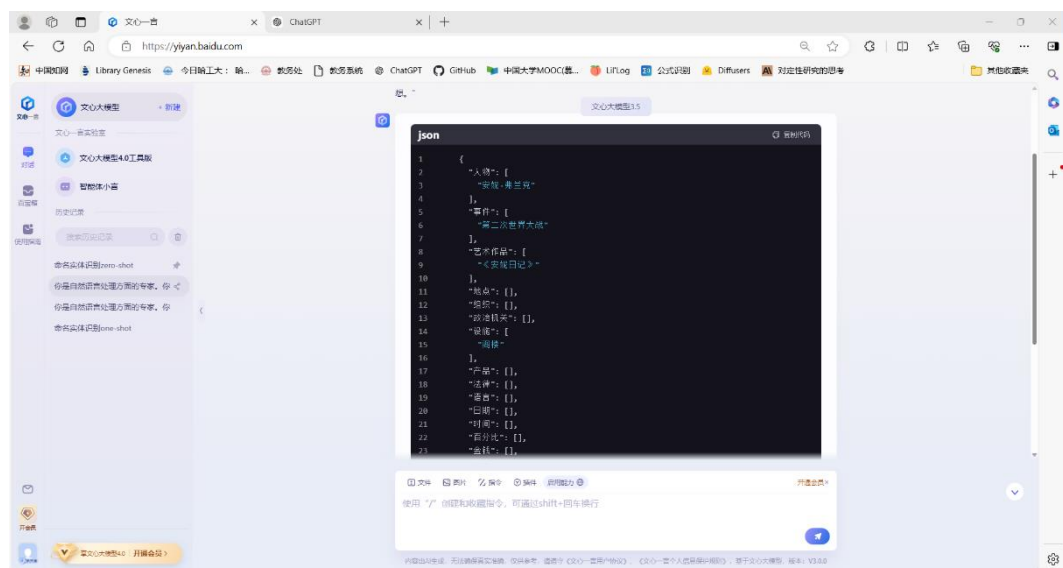
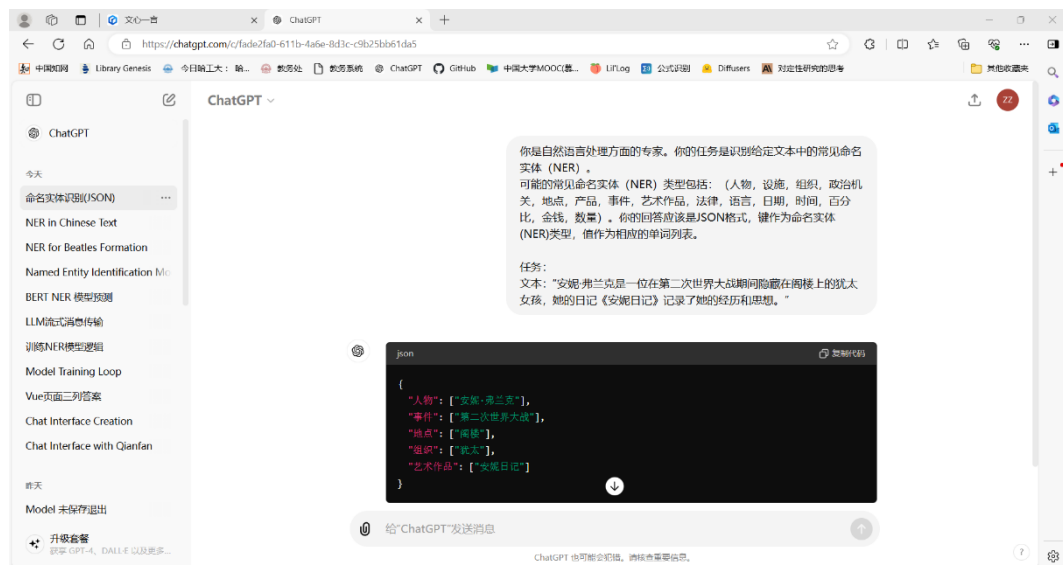
注：由于通义千问 2.5 的英文指令遵循较差，会回答“作为一个人工智能语言模型，目前我还没有学会如何回答这个问题，我会持续学习，为您提供更好的服务。”因此上述实验结果并未采用通义千问。

例句 2: 安妮·弗兰克是一位在第二次世界大战期间隐藏在阁楼上的犹太女孩, 她的日记《安妮日记》记录了她的经历和思想。

使用 GPT-3.5, 文心大模型 3.5 和通义千问 2.5 的 **One-shot** 获得结果如下:



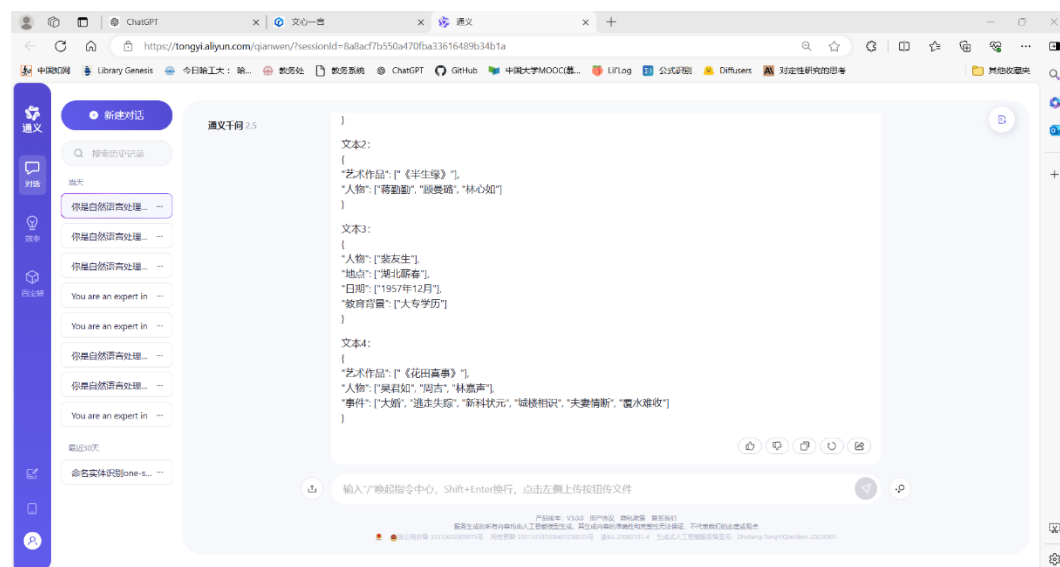
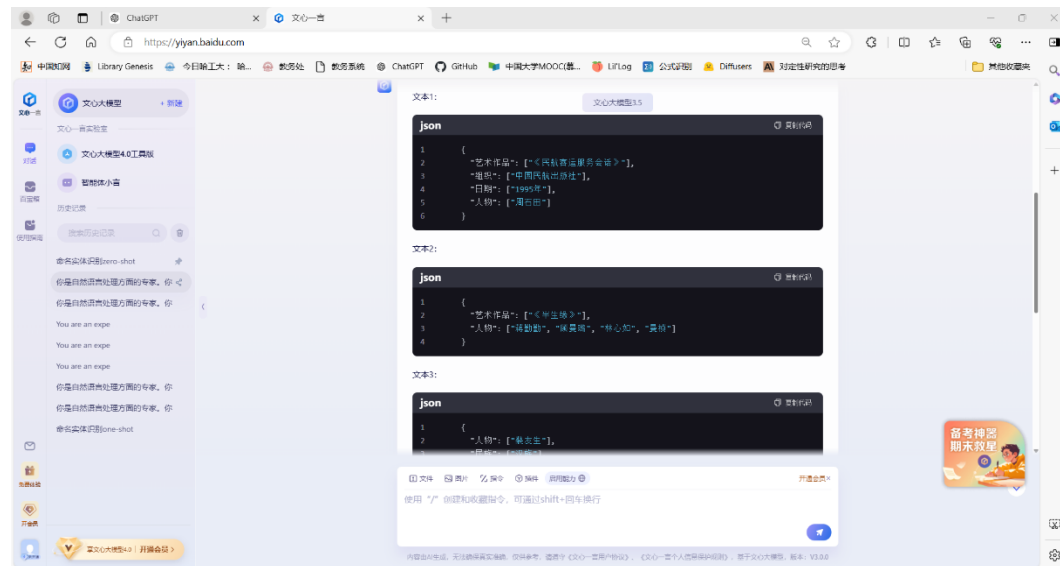
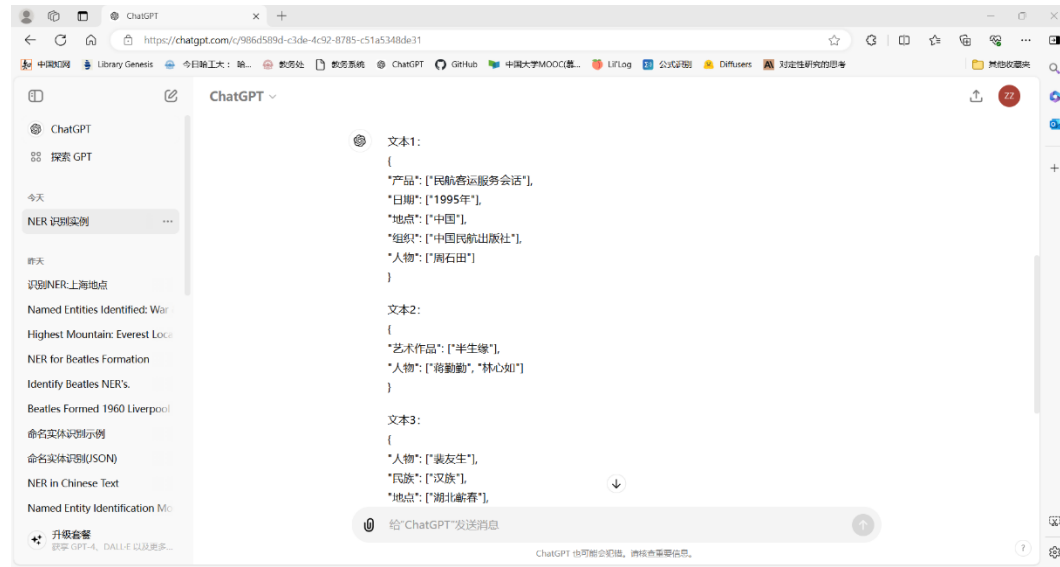
使用 GPT-3.5，文心大模型 3.5 和通义千问 2.5 的 **Zero-shot** 获得结果分别如下：



注：更多例句和实验结果请见附录第四部分。

使用第二部分中的测试用例进行测试结果如下：

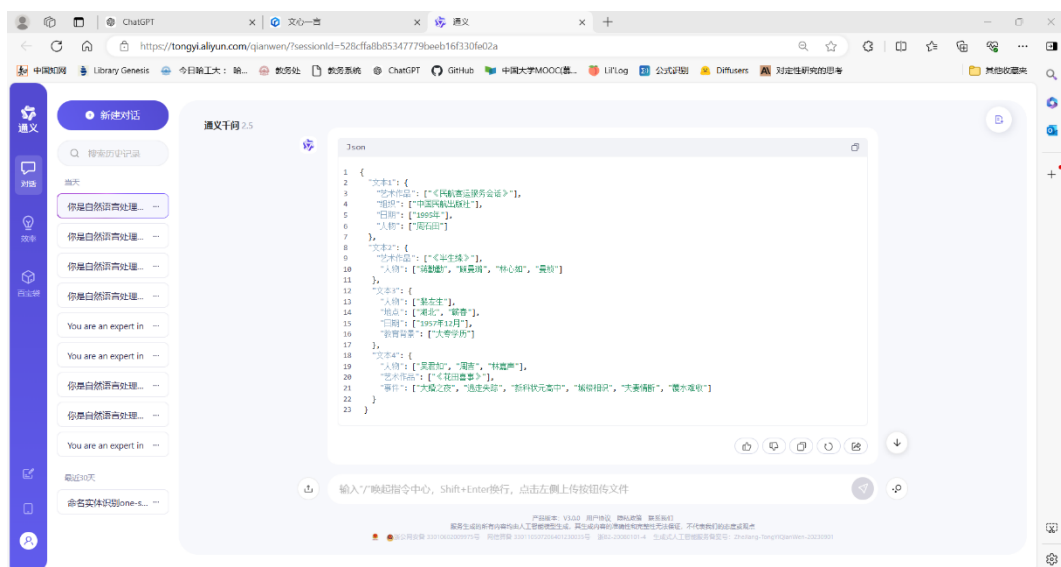
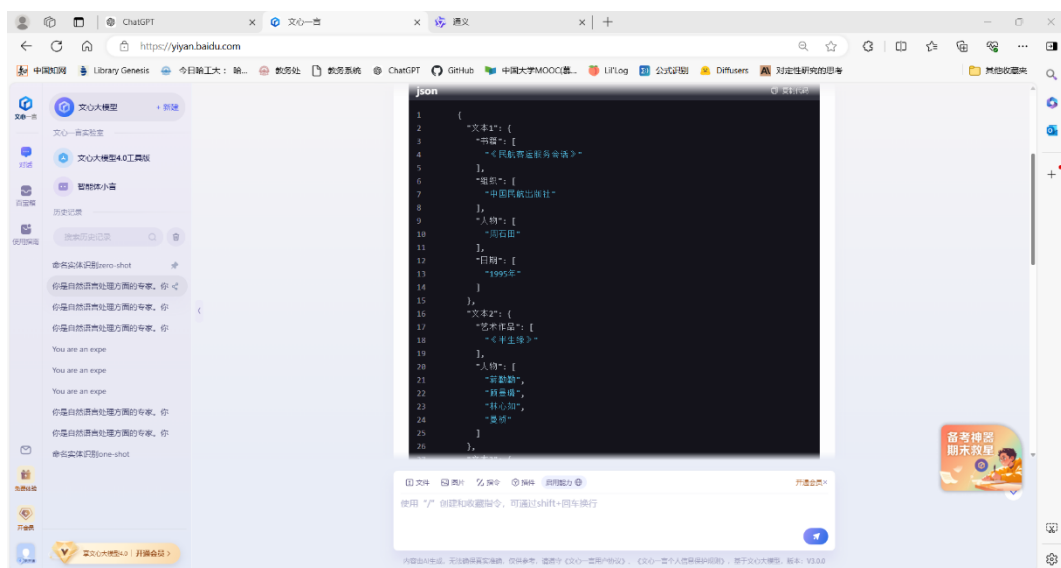
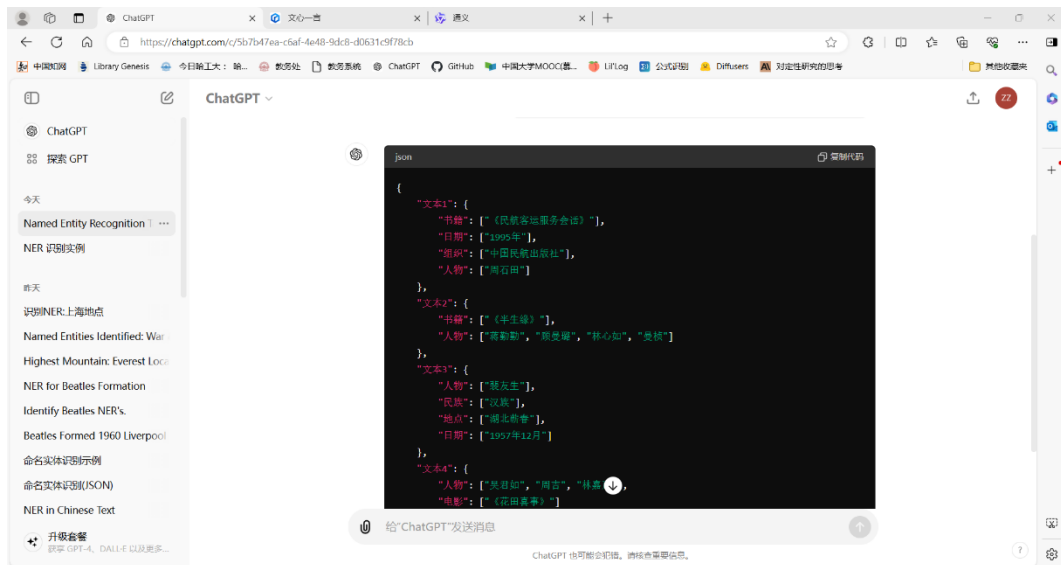
One-shot



具体结果如下表格所示：

模型	GPT-3.5	文心大模型 3.5	通义千问 2.5
回答	<p>文本 1:</p> <pre>{ "产品": ["民航客运服务 会话"], "日期": ["1995 年"], "地点": ["中国"], "组织": ["中国民航出版 社"], "人物": ["周石田"] }</pre> <p>文本 2:</p> <pre>{ "艺术作品": ["半生缘"], "人物": ["蒋勤勤", "林心 如"] }</pre> <p>文本 3:</p> <pre>{ "人物": ["裴友生"], "民族": ["汉族"], "地点": ["湖北蕲春"], "日期": ["1957 年 12 月"] }</pre> <p>文本 4:</p> <pre>{ "艺术作品": ["花田喜事"], "人物": ["吴君如", "林嘉 声"] }</pre>	<p>文本 1:</p> <pre>{ "艺术作品": ["《民航客 运服务会话》"], "组织": ["中国民航出 版社"], "日期": ["1995 年"], "人物": ["周石田"] }</pre> <p>文本 2:</p> <pre>{ "艺术作品": ["《半生 缘》"], "人物": ["蒋勤勤", "顾 曼璐", "林心如", "曼桢"] }</pre> <p>文本 3:</p> <pre>{ "人物": ["裴友生"], "民族": ["汉族"], "地点": ["湖北蕲春"], "日期": ["1957 年 12 月"], "教育水平": ["大专"] }</pre> <p>文本 4:</p> <pre>{ "艺术作品": ["《花田 喜事》"], "人物": ["吴君如", "周 吉", "林嘉声"] }</pre>	<p>文本 1:</p> <pre>{ "艺术作品": ["《民航客 运服务会话》"], "组织": ["中国民航出版 社"], "日期": ["1995 年"], "人物": ["周石田"] }</pre> <p>文本 2:</p> <pre>{ "艺术作品": ["《半生 缘》"], "人物": ["蒋勤勤", "顾 曼璐", "林心如"] }</pre> <p>文本 3:</p> <pre>{ "人物": ["裴友生"], "地点": ["湖北蕲春"], "日期": ["1957 年 12 月"], "教育背景": ["大专学历"] }</pre> <p>文本 4:</p> <pre>{ "艺术作品": ["《花田喜 事》"], "人物": ["吴君如", "周吉", "林嘉声"], "事件": ["大婚", "逃走失 踪", "新科状元", "城楼 相识", "夫妻情断", "覆 水难收"] }</pre>

Zero-shot



具体结果如下表格所示：

模型	GPT-3.5	文心大模型 3.5	通义千问 2.5
回答	<pre> ```json { "文本 1": { "书籍": ["《民航客运服务会话》"], "日期": ["1995 年"], "组织": ["中国民航出版社"], "人物": ["周石田"] }, "文本 2": { "书籍": ["《半生缘》"], "人物": ["蒋勤勤", "顾曼璐", "林心如", "曼桢"] }, "文本 3": { "人物": ["裴友生"], "民族": ["汉族"], "地点": ["湖北蕲春"], "日期": ["1957 年 12 月"] }, "文本 4": { "人物": ["吴君如", "周吉", "林嘉声"], "电影": ["《花田喜事》"] } } ``` </pre>	<pre> json { "文本 1": { "书籍": ["《民航客运服务会话》"], "组织": ["中国民航出版社"], "人物": ["周石田"], "日期": ["1995 年"] }, "文本 2": { "艺术作品": ["《半生缘》"], "人物": ["蒋勤勤", "顾曼璐", "林心如", "曼桢"] }, "文本 3": { "人物": ["裴友生"], "民族": ["汉族"], "地点": ["湖北蕲春"], "日期": ["1957 年 12 月"], "教育水平": [</pre>	<pre> ```json { "文本 1": { "艺术作品": ["《民航客运服务会话》"], "组织": ["中国民航出版社"], "日期": ["1995 年"], "人物": ["周石田"] }, "文本 2": { "艺术作品": ["《半生缘》"], "人物": ["蒋勤勤", "顾曼璐", "林心如", "曼桢"] }, "文本 3": { "人物": ["裴友生"], "地点": ["湖北", "蕲春"], "日期": ["1957 年 12 月"], "教育背景": ["大专学历"] }, "文本 4": { "人物": ["吴君如", "周吉", "林嘉声"], "艺术作品": ["《花田喜事》"], "事件": ["大婚之夜", "逃走失踪", "新科状元高中", "城楼相识", "夫妻情断", "覆水难收"] } } ``` </pre>

		"大专"] }, "文本 4": { "艺术作品": ["《花田喜事》"], "人物": ["吴君如", "周吉", "林嘉声"], "事件": ["周吉大婚"] } }	
--	--	--	--

四、附录

(1. 基于 bert 的预测结果的部分结果，详细结果见 test_entities.txt)

5. 墨写你的美: 歌曲 冷漠: 人物
6. 单身: 歌曲 Outsider: 人物 2 辑: 音乐专辑 Maestro: 音乐专辑
7. 物流作业方法: 图书作品 李培亮: 人物
8. 刘启: 人物 刘越: 人物 刘寄: 人物
9. 阎维文: 人物 刘卫星: 人物
10. 阿特·海曼: 人物 美国: 国家 美国: 国家
11. 艾布拉姆斯: 人物
12. 东汉: Text 吴汉: 人物
13. 常山公主: 人物 晋文帝: 人物 司马昭: 人物
14. 开会是一门技术活儿: 图书作品 简宁: 人物
15. 上海虹桥文化金融大楼投资有限公司: 机构 1985 年 4 月 17 日: Date
16. 李鸣生: 人物
17. 彭于晏: 人物 仙剑奇侠传: 影视作品 霍建华: 人物 基因决定我爱你: 影视作品 翻滚吧! 阿信: 影视作品
18. 佳偶天成: 图书作品 十四郎: 人物
19. 华艺照明: 机构 中山市华艺灯饰照明股份有限公司: 机构 1986 年: Date
20. 华润(集团)有限公司: 企业 宋林: 人物

.....

(2. 大模型命名实体识别 One-shot Prompt 例句)

英文

You are an expert in Natural Language Processing. Your task is to identify common Named Entities (NER) in a given text.

The possible common Named Entities (NER) types are exclusively: (person, fac, org, gpe, loc, product, event, work_of_art, law, language, date, time, percent, money, quantity).

EXAMPLE:

Text: 'In Germany, in 1440, goldsmith Johannes Gutenberg invented the movable-type printing press. His work led to an information revolution and the unprecedented mass-spread of literature throughout Europe. Modelled on the design of the existing screw presses, a single Renaissance movable-type printing press could produce up to 3,600 pages per workday.'

```
{
  "gpe": ["Germany", "Europe"],
  "date": ["1440"],
  "person": ["Johannes Gutenberg"],
  "product": ["movable-type printing press"],
  "event": ["Renaissance"],
  "quantity": ["3,600 pages"],
  "time": ["workday"]
}
```

TASK:

Text: The Beatles were an English rock band formed in Liverpool in 1960, comprising John Lennon, Paul McCartney, George Harrison, and Ringo Starr.

中文

你是自然语言处理方面的专家。你的任务是识别给定文本中的常见命名实体（NER）。

可能的常见命名实体（NER）类型包括：（人物，设施，组织，政治机关，地点，产品，事件，艺术作品，法律，语言，日期，时间，百分比，金钱，数量）。

示例：

文本：“在美国的太空探索历史上，阿波罗 11 号任务标志着人类首次成功登月。1969 年 7 月 20 日，尼尔·阿姆斯特朗和巴兹·奥尔德林成为第一批登上月球的人类。”

```
{
  "政治机关": ["美国"],
  "事件": ["阿波罗 11 号任务", "登月"],
  "日期": ["1969 年 7 月 20 日"],
```

"人物": ["尼尔·阿姆斯特朗", "巴兹·奥尔德林"]
}

任务:

文本: “安妮·弗兰克是一位在第二次世界大战期间隐藏在阁楼上的犹太女孩, 她的日记《安妮日记》记录了她的经历和思想。”

(3. 大模型命名实体识别 Zero-shot Prompt 例句-英文)

英文

You are an expert in Natural Language Processing. Your task is to identify common Named Entities (NER) in a given text.

The possible common Named Entities (NER) types are exclusively: (person, fac, org, gpe, loc, product, event, work_of_art, law, language, date, time, percent, money, quantity). Your response should be in JSON format, with keys as Named Entities (NER) types and values as the corresponding list of words.

TASK:

Text: The Beatles were an English rock band formed in Liverpool in 1960, comprising John Lennon, Paul McCartney, George Harrison, and Ringo Starr.

中文

你是自然语言处理方面的专家。你的任务是识别给定文本中的常见命名实体 (NER)。

可能的常见命名实体 (NER) 类型包括: (人物, 设施, 组织, 政治机关, 地点, 产品, 事件, 艺术作品, 法律, 语言, 日期, 时间, 百分比, 金钱, 数量)。你的回答应该是 JSON 格式, 键作为命名实体 (NER) 类型, 值作为相应的单词列表。

任务:

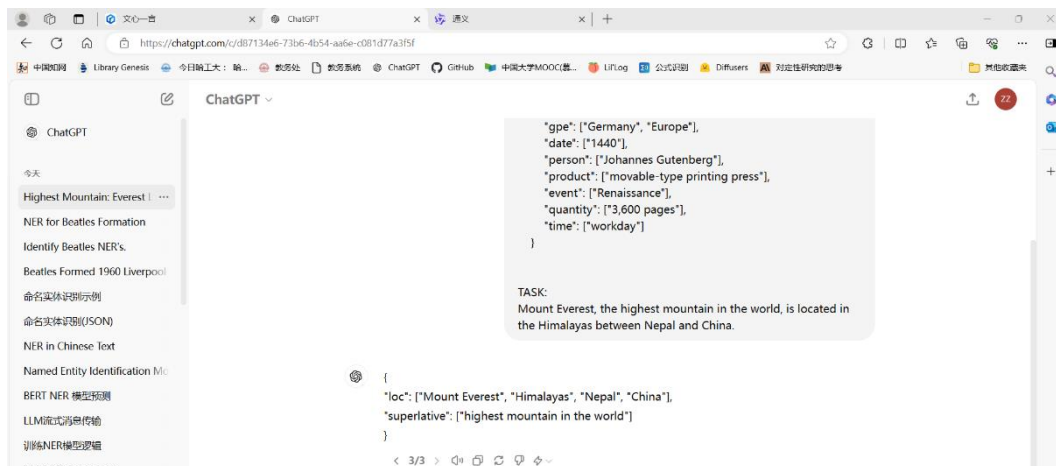
文本: “安妮·弗兰克是一位在第二次世界大战期间隐藏在阁楼上的犹太女孩, 她的日记《安妮日记》记录了她的经历和思想。”

(4. 更多基于大模型的命名实体识别的实验结果)

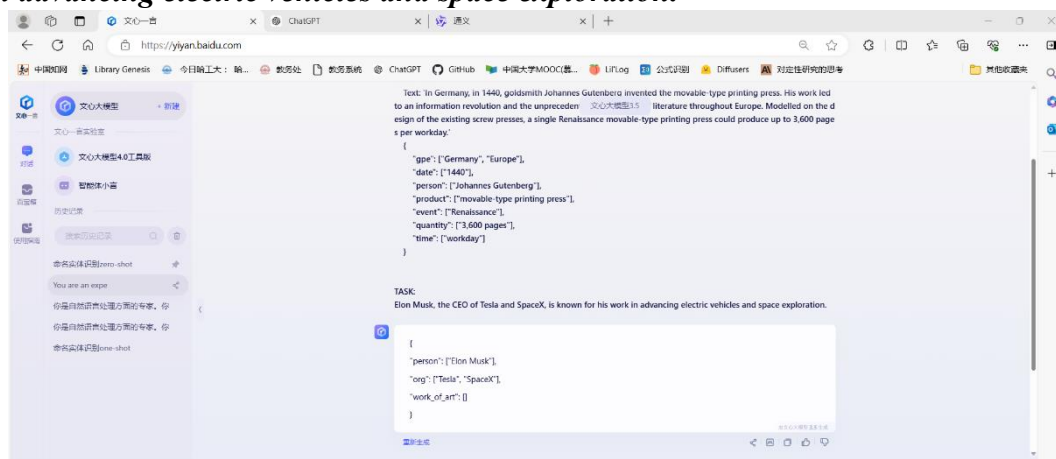
由于各个模型结果几乎一致, 因此下述只截取其中一种模型的结果(模型包括 GPT-3.5, 文心大模型 3.5 和通义千问 2.5)。

One-shot 部分

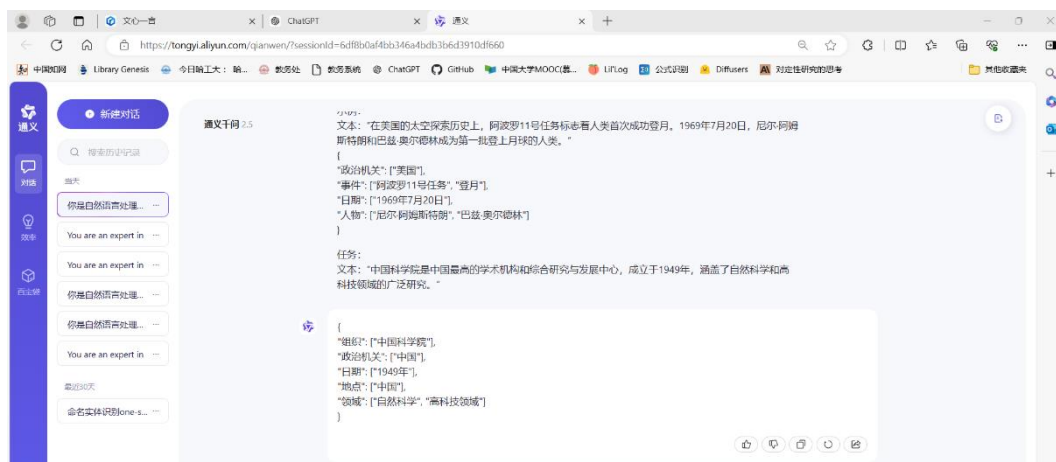
补充例句1: *Mount Everest, the highest mountain in the world, is located in the Himalayas between Nepal and China.*



补充例句2: *Elon Musk, the CEO of Tesla and SpaceX, is known for his work in advancing electric vehicles and space exploration.*



补充例句3: *中国科学院是中国最高的学术机构和综合研究与发展中心，成立于1949年，涵盖了自然科学和高科技领域的广泛研究。*

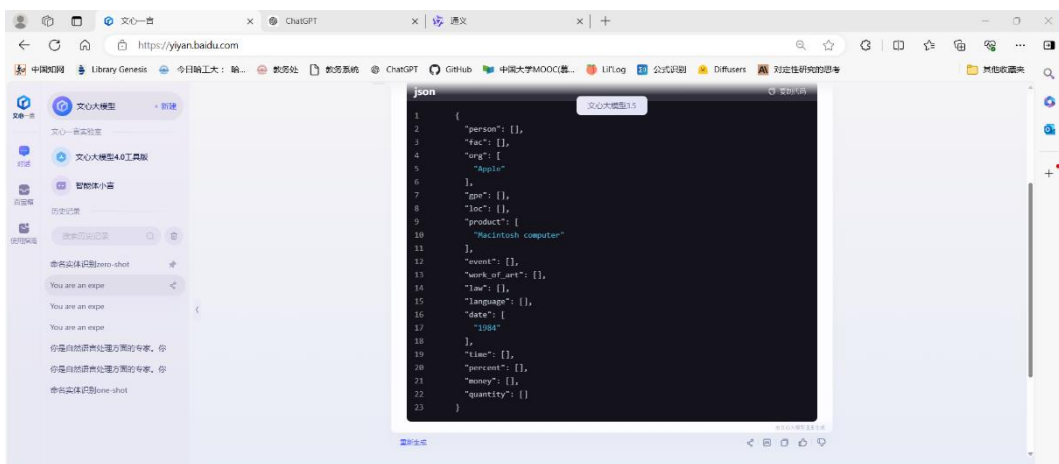


补充例句4:《清明上河图》是北宋画家张择端创作的一幅长卷风俗画,描绘了汴京(今开封)清明时节的繁荣景象,被誉为中国古代绘画的瑰宝。

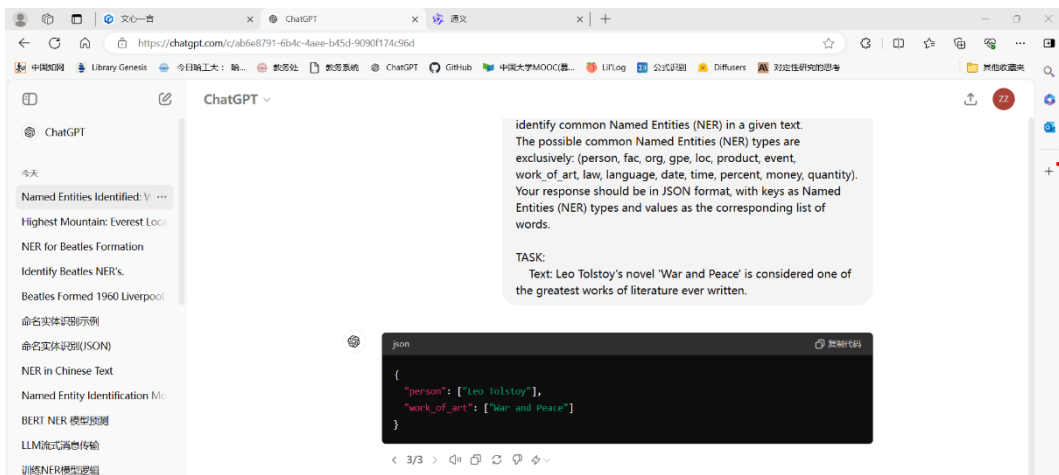


Zero-shot 部分

补充例句1: *The release of the first Macintosh computer by Apple in 1984 revolutionized the personal computer industry.*



补充例句2: *Leo Tolstoy's novel 'War and Peace' is considered one of the greatest works of literature ever written.*



补充例句3: 上海是中国最大的城市之一, 以其现代化的天际线和繁忙的金融区而闻名, 同时也保留了众多历史悠久的地标, 如外滩和豫园。



补充例句4: 珠穆朗玛峰是世界最高的山峰, 位于喜马拉雅山脉的中部, 横跨中国和尼泊尔的边界, 是许多登山者梦想征服的圣地。

