“自然语言处理”实验报告

实验 2：命名实体识别

姓名：王艺丹　　学号：1190201303

Email：1340258795@qq.com

1. 实验概述

本次实验学习使用 HMM、ME、CRF 和深度学习等不同的命名实体识别方法，并在两个不用的数据集上进行实验。实验需要用到百度 AI Studio 平台和华为云计算平台。

2. 实验目标

- 通过不同方法的结果对比，掌握不同实体识别方法的优缺点。
- 通过对不同数据集的使用，掌握命名实体识别需要的数据预处理、模型训练、模型测试和评价方法。
- 通过对百度 AI Studio 平台和华为云平台的使用，了解国产主流人工智能平台提供的学习资源和计算资源，并能使用这些平台完成特定的自然语言处理任务。
- 通过实验报告的撰写，提高分析能力、写作能力和表达能力。

3. 命名实体识别的评价

1）命名实体识别的评价指标及计算公式

命名体识别的评价指标主要有精确率、召回率、F1 值等指标，可以分为词级别与实体级别，计算公式如下：

精确率：

$$Precision = \frac{识别出的正确词/实体数}{识别出的总词/实体数} = \frac{TP}{TP + FP}$$

召回率：

$$Recall = \frac{识别出的正确词/实体数}{正确的总词/实体数} = \frac{TP}{TP + FN}$$

F1 值：

$$F = \frac{2PR}{P + R}$$

其中对于所有类实体总的PRF有两种计算方式：按类别平均和按样本平均[1]

- 宏平均：分别计算单实体评估指标，再求整体的平均值（将每个实体类别视为平等）

- 微平均：对整体数据求评估指标（将每个实体视为平等）

本实验采用**微平均。**

2）自己实现的实体级命名实体识别的评价程序的实现思路（用文字、流程图及伪代码进行说明）

主要思路：

编写类$calEntityPRF$，需要传入$test_tag_lists$与$pred_tag_lists$进行初始实例化，主要有三个功能函数：$trans2entity$，cal_entity_PRF与辅助输出函数$show_scores$。其中$trans2entity$函数实现对标签序列进行实体识别并转为位置下标区间集合计入字典；cal_entity_PRF函数利用集合的交运算等进行实体级别的P、R、F的计算；$show_scores$函数实现将结果按词级别的输出格式打印输出，下面为各函数的具体思想

a) $\boldsymbol{\_\_init\_\_(self,\ test\_tag\_lists, pred\_tag\_lists)}$

调用 $self.cal\_entity\_PRF(test\_tag\_lists, pred\_tag\_lists)$

对 $self.P, self.R, self.F, self.support$ 进行初始化

b) $\boldsymbol{trans2entity}$

$param$: $list\ tag\_list$

$return$: $dict\ num$

主要算法流程：

[1] 对单句话的标签序列按位置下标 i 进行遍历

[2] 判断标记类型与实体类型$type$

i. 为 'O'，令$flag = 0$，继续遍历

ii. 为 'B'，利用pre记录实体类型，用$start$记录起始位置下标，更新$flag$为1，继续遍历

iii. 为 'S'，将区间$(i,\ i)$加入$num[type]$中，令$flag = 0$，继续遍历

iv. 为 'M' 且$flag == 1$，判断当前类型是否与pre一致，一致则继续遍历；否则更新pre为空，$flag$为0，继续遍历

v. 为 'E' 且$flag == 1$，更新$flag$为0，pre为空，判断当前类型是否与pre一致，一致则将区间$(start,\ i)$加入$num[type]$中，继续遍历；否则不加入集合，继续遍历

[3] 遍历结束，返回存储各类型实体对应位置区间集合的字典num

c) $\boldsymbol{cal\_entity\_PRF}$

$param$: $list\ test\_tag\_lists$ #标准答案各句话标签列表构成的列表

$list\ pred\_tag\_lists$ #预测结果各句话标签列表构成的列表

$return$: $dict\ P$, $dict\ \ R$, $dict\ \ F$, $dict\ support$ # 记录各实体类型对应指标值的字典，其中$support$记录正确的实体个数

主要算法流程：

[1] $assert\ len(pred\_tag\_lists)\ ==\ len(test\_tag\_lists)$

[2] 初始化各字典的$avg/total$为 0；

初始化标准答案实体类型$tags$与tt为空集合；

初始化TP为空字典便于后续计算

[3] 按下标 i 对列表进行遍历，记

$pred\_sets = pred\_tag\_lists[i]$,

$test\_sets = test\_tag\_lists[i]$;

$setP = trans2entity(pred\_sets)$,

$setR = trans2entity(test\_sets)$

得到单句话对应的各类型对应的位置区间集合；

初始化$pred$与$support$为空字典；

更新tt为空，用于后续判断预测结果与实际结果的出入

[4] 对$tag\ in\ setP.keys()$遍历，将tag加入集合$tags$与tt中；

利用集合的交运算计算$setP$与$setR$的交集，交集的元素个数即为正确预测出的实体数TP，累加至$TP[tag]$中；

$len(setP)$ 即为识别出的实体总数$TP + FP$，暂时将$len(setP)$累加计入至$pred[type]$中；

$len(setR)$ 即为正确的实体总数$TP + FN$，将$len(setR)$累加计入至$support[type]$中；

[5] 对 $tag\ in\ pred_sets.keys()$遍历，判断其 是否在$tt$中，若不在，则代表预测出了不应存在的预测实体，精确率计算中的分母缺失该部分，则将$len(pred_sets[tag])$累加至$pred[tag]$中

[6] 对$tag\ in\ tags$进行遍历，将各指标结果累加只各指标'$avg/total$'对应的$value$中；按公式计算各指标：

$$P[tag] = \frac{TP[tag]}{pred[tag]}$$

$$R[tag] = \frac{TP[tag]}{support[tag]}$$

$$F[tag] = 2 * P[tag] * \frac{R[tag]}{P[tag] + R[tag] + e^{-10}}\ \#防止P与R同时为0$$

[7] 令$tag =' avg/total'$, 同理按上述公式计算各指标均值

[8] 返回$P,\ R,\ F,\ support$

d) ***show_scores***

具体思想与$evaluating.py$中的输出思想一致，不做赘述

3）自己实现的评价程序的调用方法（含参数说明）

首先得到测试数据实际标记序列列表$test_tag_lists$，再利用相应模型进行预测得到预测结果标记序列列表$pred_tag_lists$，对类$calEntityPRF$进行实例化，之后对示例化的对象调用$show_scores()$对结果进行打印输出，例:

```
"""计算并打印实体级别HMM模型的精确率、召回率、F值"""
entityPRF_HMM = calEntityPRF(test_tag_lists_clue, pred_tag_lists_HMM)
entityPRF_HMM.show_scores()
```

4）对 HMM 模型在 ner_char_data 目录下的 test.txt 文件上的识别结果分别按词级别和实体级别进行评价，给出分类别的 P、R、F 值和总体的 P、R、F 值，并分析两种评价方式结果差别的原因。

词级别：

	precision	recall	f1-score	support
B-NAME	0.9800	0.8750	0.9245	112
M-NAME	0.9459	0.8537	0.8974	82
E-NAME	0.9000	0.8036	0.8491	112
O	0.9568	0.9177	0.9369	5190
B-PRO	0.5581	0.7273	0.6316	33
E-PRO	0.6512	0.8485	0.7368	33
B-EDU	0.9000	0.9643	0.9310	112
E-EDU	0.9167	0.9821	0.9483	112
B-TITLE	0.8811	0.8925	0.8867	772
M-TITLE	0.9038	0.8751	0.8892	1922
E-TITLE	0.9514	0.9637	0.9575	772
B-ORG	0.8422	0.8879	0.8644	553
M-ORG	0.9002	0.9327	0.9162	4325
E-ORG	0.8262	0.8680	0.8466	553
B-CONT	0.9655	1.0000	0.9825	28
M-CONT	0.9815	1.0000	0.9907	53
E-CONT	0.9655	1.0000	0.9825	28
M-EDU	0.9348	0.9609	0.9477	179
B-RACE	1.0000	0.9286	0.9630	14
E-RACE	1.0000	0.9286	0.9630	14
B-LOC	0.3333	0.3333	0.3333	6
M-LOC	0.5833	0.3333	0.4242	21
E-LOC	0.5000	0.5000	0.5000	6
M-PRO	0.4490	0.6471	0.5301	68
avg/total	0.9149	0.9122	0.9130	15100

实体级别：

	precision	recall	f1-score	support
LOC	0.3333	0.3333	0.3333	6
ORG	0.7543	0.7884	0.7710	553
NAME	0.8491	0.8036	0.8257	112
EDU	0.8917	0.9554	0.9224	112
CONT	0.9655	1.0000	0.9825	28
PRO	0.5581	0.7273	0.6316	33
TITLE	0.8747	0.8860	0.8803	772
RACE	0.8667	0.9286	0.8966	14
avg/total	0.8243	0.8491	0.8365	1630

差异：

可以看到，词级别的分析在各评价指标上绝大部分均高于实体级的评价结果约 10%。

原因：

- ✓ **评价松弛度不同：**
  词级别的评价方法可以看作是要求更为松弛的评价方法。在 NER 任务中，最终的目的是抽取出一个实体，而词标注仅仅是模型输出的一个

结果。从模型输出结果到最终形成的 NER 任务的结果，还需要经过一个词序列处理的过程。可见实体级别的评价比词级别的评价更严格。

- ✓ **评价粒度不同：**
  实体级评价的粒度大，而词语级评价粒度小，也就是说，词语级更能捕捉小粒度上的正确结果，因此导致了最终的结果上，词语级评价方法获得的指标更高。实体级评价中正确的结果在词级别中一定正确，但在词级别评价中正确的结果，在实体级评价中不一定正确，这也就造成了词级别评价的方法，正确的概率更高。

4.基于最大熵模型的实体识别

1）最大熵模型原理的简单介绍

最大熵原理是概率模型学习的一个准则。最大熵原理认为，学习概率模型时，在所有可能的概率模型(分布中，熵最大的模型是最好的模型。通常用约束条件来确定概率模型的集合，所以，最大熵原理也可以表述为在满足约束条件的模型集合中选取熵最大的模型。

直观地，最大熵原理认为要选择的概率模型首先必须满足已有的事实，即约束条件。在没有更多信息的情况下，那些不确定的部分都是“等可能的”．最大熵原理通过熵的最大化来表示等可能性。“等可能”不容易操作，而熵则是一个可优化的数值指标。[2]

最大熵模型是一种有监督的概率模型，能以满足限定条件下的熵值最大化为准则，对各种不同类型的特征训练给出一组对应的权值，再通过线性组合，将其整合到统一的模型中，求解公式如下：

$$P(y|x) = \frac{1}{z(x)} \sum_i \lambda_i f_i(x, y)$$

$$z(x) = \sum_y \exp\left(\sum \lambda_i f_i(x, y)\right)$$

2）描述自己实现的基于最大熵模型的实体识别系统

最大熵模型是一个多分类模型，而中文命名实体识别任务也可以转化为多分类任务。即对每一词 / 字(单元)给一个标记。我们采用 B-M-E-S-O 标记体系，其中，B 表示命名实体的起始单元，M 表示命名实体中的其它单元，E 表示命名实体中的结束单元，如果一个单元单独成为一个实体，则用 S 表示，O 表示各命名实体之外的单元。而将最大熵模型应用到 NER 任务的过程就是对每一个字做多分类的过程。[3]

利用$sklearn$中已有的$LogisticRegression$进行实现，将$train_word_lists$中每一句话进行进行特征提取，利用$DictVectorizer$中的$fit_transform$函数进行特征抽取，再结合$train_tag_lists$利用$LogisticRegression$自带的 fit 函数进行训练；将待预测的句子$test_word_lists$中的每一句话利用$DictVectorizer$中的

$transform$函数，利用已有特征向量进行转换，再利用$LogisticRegression$的$predict$函数进行预测，并将结果转换为列表。

3）详细说明自己实现的基于最大熵模型的实体识别系统利用的特征

使用的特征如下：

```
# 使用的特征：
# 前一个词，当前词，后一个词，
# 前一个词+当前词， 当前词+后一个词
features = {
    'w': word,
    'w-1': prev_word,
    'w+1': next_word,
    'w-1:w': prev_word+word,
    'w:w+1': word+next_word,
    'w-1:w:w+1':prev_word+word+next_word,
}
```

原因主要为：

包含上下文词汇信息，包括当前词，前一个词，下一个词，并将三者的组合也作为一个特征输入；同时体现词在句子中的位置信息

4）评价自己实现的基于最大熵模型的实体识别系统的效果

利用 ner_char_data 目录下的 train.txt 文件训练模型，在 test.txt 文件上进行测试。对测试结果按词级别和实体级别进行评价，给出分类别的 P、R、F 值和总体的 P、R、F 值。

```
"""利用ner_char_data目录下的train.txt文件训练模型"""
ME_test = ME()
ME_test.train(train_word_lists, train_tag_lists)
```

```
"""在test.txt文件上进行测试"""
with open('./work/test_ME.txt', 'w', encoding='utf-8') as me_test:
    pred_test_tag_lists_ME = []
    for word_list in test_word_lists:
        pred_tag_list = ME_test.predict(word_list)
        pred_test_tag_lists_ME.append(pred_tag_list)
        for i in range(len(word_list)):
            me_test.write(word_list[i] + ' ' + pred_tag_list[i] + '\n')
        me_test.write('\n')
```

词级别：

```
# 词级别
metrics_me = Metrics(test_tag_lists, pred_test_tag_lists_ME, remove_O=False)
metrics_me.report_scores()
```

	precision	recall	f1-score	support
B-NAME	0.8203	0.9375	0.8750	112
M-NAME	0.9400	0.5732	0.7121	82
E-NAME	0.9646	0.9732	0.9689	112
O	0.9612	0.9443	0.9527	5190
B-PRO	0.8571	0.5455	0.6667	33
E-PRO	0.7750	0.9394	0.8493	33
B-EDU	0.9145	0.9554	0.9345	112
E-EDU	0.9646	0.9732	0.9689	112
B-TITLE	0.9114	0.8795	0.8952	772
M-TITLE	0.9063	0.8652	0.8853	1922
E-TITLE	0.9896	0.9896	0.9896	772
B-ORG	0.8881	0.8608	0.8742	553
M-ORG	0.8931	0.9450	0.9183	4325
E-ORG	0.9052	0.8807	0.8928	553
B-CONT	0.6000	0.9643	0.7397	28
M-CONT	1.0000	1.0000	1.0000	53
E-CONT	1.0000	0.9643	0.9818	28
M-EDU	0.9556	0.9609	0.9582	179
B-RACE	1.0000	0.9286	0.9630	14
E-RACE	1.0000	1.0000	1.0000	14
B-LOC	0.0000	0.0000	0.0000	6
M-LOC	0.7500	0.1429	0.2400	21
E-LOC	1.0000	0.8333	0.9091	6
M-PRO	0.5676	0.6176	0.5915	68
avg/total	0.9239	0.9231	0.9224	15100

实体级别：

```
# 实体级别
PRF_ME_test = calEntityPRF(test_tag_lists, pred_test_tag_lists_ME)
PRF_ME_test.show_scores()
```

	precision	recall	f1-score	support
ORG	0.9297	0.6456	0.7620	553
TITLE	0.9636	0.8238	0.8883	772
NAME	1.0000	0.6071	0.7556	112
LOC	0.0000	0.0000	0.0000	6
RACE	1.0000	0.9286	0.9630	14
CONT	1.0000	0.9286	0.9630	28
PRO	1.0000	0.4242	0.5957	33
EDU	0.9813	0.9375	0.9589	112
avg/total	0.9583	0.7479	0.8401	1630

5. HMM、ME 与 CRF 的效果对比

1）说明如何利用给定的 ner_clue_data 目录下的训练数据文件 train.txt 分别训练 HMM、ME 和 CRF 模型并使用 dev.txt 文件里的数据来测试三个模型。

[1] 首先仿照已有函数*data_build*的数据处理模式，编写读入 ner_clue_data 目录下 json 数据的函数 *data_build_clue*，并调用，获得 *train_word_lists_clue*，*train_tag_lists_clue*，*word2id_clue*，*tag2id_clue*，*test_word_lists_clue*, *test_tag_lists_clue*

```
# 在这里实现利用新数据重新训练和测试HMM、ME和CRF模型
"""读入ner_clue_data下train.txt数据"""
train_word_lists_clue, train_tag_lists_clue, word2id_clue, tag2id_clue = data_build_clue(file_name="train.txt", make_vocab=True)
test_word_lists_clue, test_tag_lists_clue = data_build_clue(file_name="dev.txt", make_vocab=False)
```

[2] 将 HMM 单元代码块内的代码改写为类，传入相应参数训练构建 HMM_clue，使用类方法对 test_word_lists_clue 进行预测，得到 pred_tag_lists_HMM，并将结果写入 work 目录下的 dev_HMM.txt

```
""" 用ner_clue_data目录下train.txt中的数据训练HMM，并对dev.txt进行预测，并将结果写入work文件夹下的dev_HMM.txt"""
HMM_clue = HMM(train_word_lists_clue, train_tag_lists_clue, word2id_clue, tag2id_clue)
with open('./work/dev_HMM.txt', 'w', encoding='utf-8') as hmm:
    pred_tag_lists_HMM = []
    for word_list in test_word_lists_clue:
        pred_tag_list = HMM_clue.viterbi(word_list, word2id_clue, tag2id_clue)
        pred_tag_lists_HMM.append(pred_tag_list)
        for i in range(len(word_list)):
            hmm.write(word_list[i] + ' ' + pred_tag_list[i] + '\n')
        hmm.write('\n')
```

[3] 类似地，将相应参数传入 ME 类构建 ME_clue，并利用类方法 predict 对 $test_word_lists_clue$进行预测，得到$pred_tag_lists_ME$，并将结果写入 work 目录下的 dev_ME.txt

```
"""用ner_clue_data目录下train.txt中的数据训练最大熵ME模型"""
ME_clue = ME()
ME_clue.train(train_word_lists_clue, train_tag_lists_clue)
```

```
"""用ME模型预测ner_clue_data目录下dev.txt中的数据并将结果写入work文件夹下的dev_ME.txt"""
with open('./work/dev_ME.txt', 'w', encoding='utf-8') as me:
    pred_tag_lists_ME = []
    for word_list in test_word_lists_clue:
        pred_tag_list = ME_clue.predict(word_list)
        pred_tag_lists_ME.append(pred_tag_list)
        for i in range(len(word_list)):
            me.write(word_list[i] + ' ' + pred_tag_list[i] + '\n')
        me.write('\n')
```

[4] 类似地，将相应参数传入 CRF 类构建 CRF_clue，并利用类方法 predict 对 $test_word_lists_clue$进行预测，得到$pred_tag_lists_CRF$，并将结果写入 work 目录下的 dev_CRF.txt

```
"""用ner_clue_data目录下train.txt中的数据训练CRF模型"""
CRF_clue = CRFModel()
CRF_clue.train(train_word_lists_clue, train_tag_lists_clue)
```

```
"""用CRF模型预测ner_clue_data目录下dev.txt中的数据并将结果写入work文件夹下的dev_CRF.txt"""
pred_tag_lists_CRF = CRF_clue.test(test_word_lists_clue)

with open('./work/dev_CRF.txt', 'w', encoding='utf-8') as crf:
    for i in range(len(test_word_lists_clue)):
        word_list = test_word_lists_clue[i]
        pred_tag_list = pred_tag_lists_CRF[i]
        for k in range(len(word_list)):
            crf.write(word_list[k] + ' ' + pred_tag_list[k] + '\n')
        crf.write('\n')
```

2）利用表格列出每个模型对应的每种实体以及总体的 Precision、Recall 和 F1 值。分析 3 种模型的结果差异，并给出可能的原因。

HMM	Recall	F1-Score	support	support
COMPANY	0.5124	0.5476	0.5294	378
ORGANIZATION	0.5237	0.5422	0.5328	367
GAME	0.6906	0.7186	0.7043	295
ADDRESS	0.3668	0.3727	0.3697	373

POSITION	0.6413	0.6605	0.6507	433
NAME	0.5802	0.6065	0.5931	465
GOVERNMENT	0.5052	0.5951	0.5465	247
MOVIE	0.5449	0.6424	0.5897	151
BOOK	0.6263	0.4026	0.4901	154
SCENE	0.4439	0.4354	0.4396	209
avg/total	0.5424	0.5605	0.5513	3072

ME	Precision	Recall	F1-Score	support
COMPANY	0.8589	0.3704	0.5176	378
ORGANIZATION	0.8824	0.5722	0.6942	367
GAME	0.8810	0.6271	0.7327	295
ADDRESS	0.6875	0.2064	0.3175	373
POSITION	0.8927	0.5958	0.7147	433
NAME	0.8929	0.4839	0.6276	465
GOVERNMENT	0.8681	0.3198	0.4675	247
MOVIE	0.8837	0.2517	0.3918	151
BOOK	0.8261	0.2468	0.3800	154
SCENE	0.7708	0.1770	0.2879	209
avg/total	0.8626	0.4189	0.5640	3072

CRF	Recall	F1-Score	support	support
COMPANY	0.7955	0.7407	0.7671	378
ORGANIZATION	0.8062	0.7139	0.7552	367
GAME	0.8161	0.8271	0.8215	295
ADDRESS	0.5932	0.4692	0.5240	373
POSITION	0.8191	0.7113	0.7614	433
NAME	0.8082	0.7247	0.7642	465
GOVERNMENT	0.7846	0.7814	0.7830	247
MOVIE	0.6818	0.6954	0.6885	151
BOOK	0.7869	0.6234	0.6957	154
SCENE	0.6733	0.4833	0.5627	209
avg/total	0.7679	0.6839	0.7235	3072

差异:

易知:

性能比较	Precision	Recall	F1-Score

实体级别	ME>CRF>HMM	CRF>HMM>ME	CRF>ME>HMM

总体来看，CRF 的表现最优，ME 其次，HMM 最后。但 ME 模型在召回率上有时表现比 HMM 差。

原因：

在 HMM 和 CRF 的比较上，由于 HMM 可以看作一种特殊形式的 CRF[4]，即在 CRF 中，如果将前一时刻和当前时刻的标签构成特征函数，并且将权重看作 HMM 中的转移概率，用当前时刻的标签和当前时刻对应的词构成的特征函数，将权重看作 HMM 中的发射概率，此时 CRF 和 HMM 模型的表达能力相同。但在实际的应用中，由于：

- ✧ CRF 利用了更丰富的特征信息
- ✧ CRF 中“权重”的表达能力强于 HMM 中“概率”的表达能力
- ✧ CRF 的概率归一化更合理，解决了 HMM 中 label bias 的问题

而正是因为表达了更丰富的信息，所以 CRF 模型的表现效果实际优于 HMM。

另外，由于 ME 使用了相当局限的局部信息，本次实验中仅利用了上下文信息，而只能达到局部最优解，但 CRF 可以求解全局最优解，表现结果更好。

但由于 ME 抽取出的实体数目小于 CRF，所以导致在某些情况下，ME 模型得到的准确率高于 CRF 模型。除此之外，ME 模型不引入新的假设的性质也带来了准确率的提高。但是这也导致了 ME 模型的召回率更低。从 F1-Score 上来看，CRF 优于 HMM。

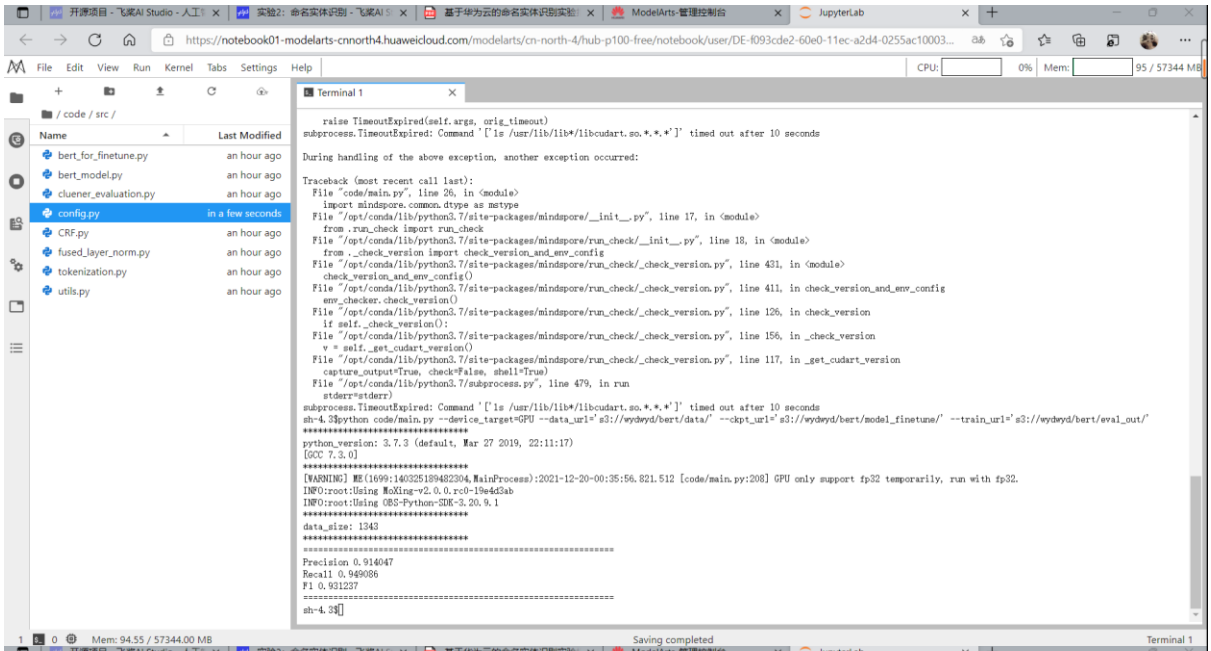
在 ME 和 HMM 的比较上，HMM 实际上表达了一个概率有向图，仅利用了前向的信息，而 ME 使用了更丰富的上下文信息，二者的局限性都是仅求解了局部最优解。在总体的性能上，由于 ME 利用的信息更丰富，所以 ME 模型的效果优于 HMM 模型。

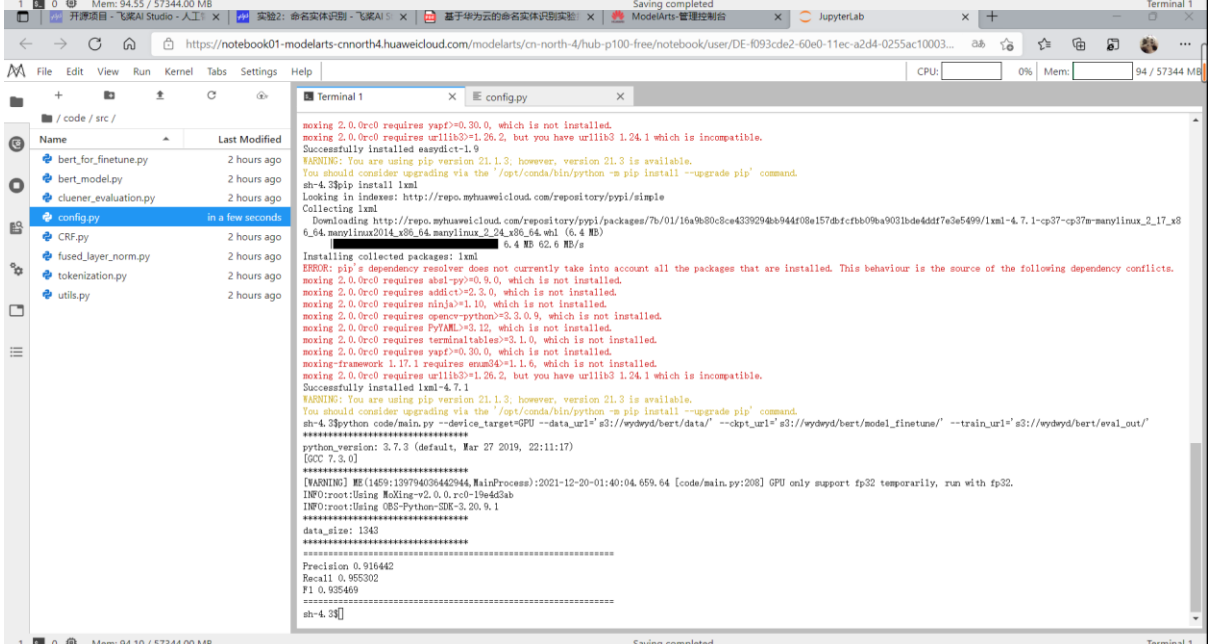
所以总体来说，模型表现效果为：CRF>ME>HMM

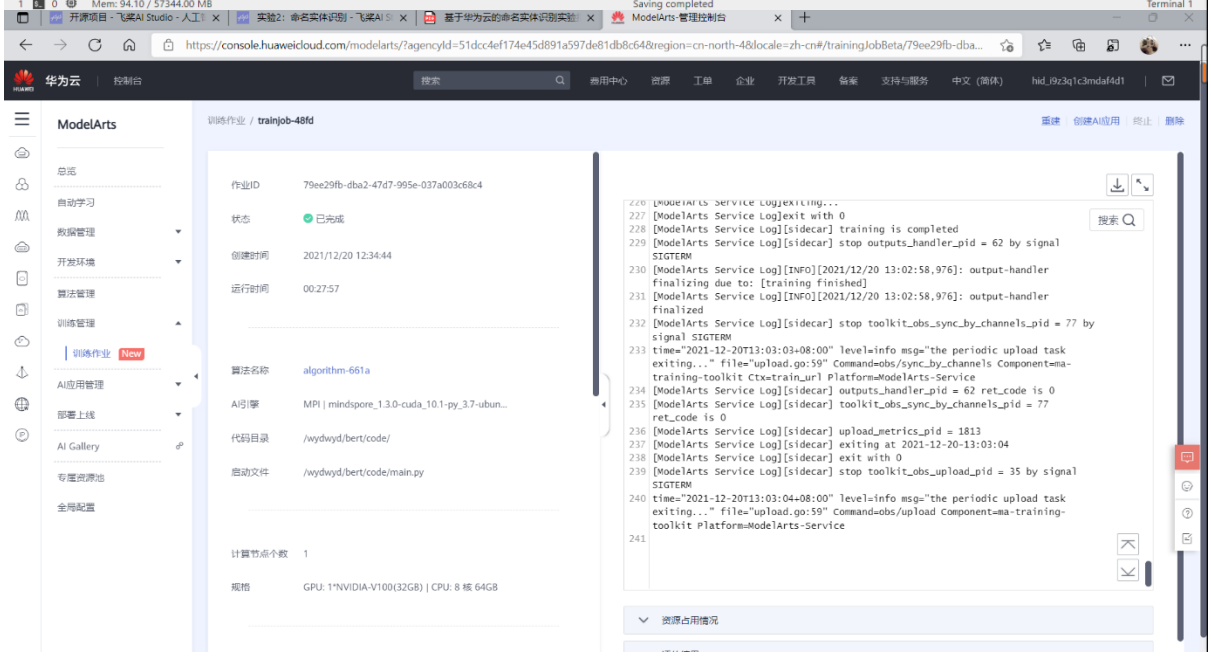
6. 在华为云上的计算资源进行命名实体识别

1）至少 2 张包含自己华为云账号信息的运行过程及结果截图，证明自己完成了利用华为云上的计算资源进行命名实体识别的任务。

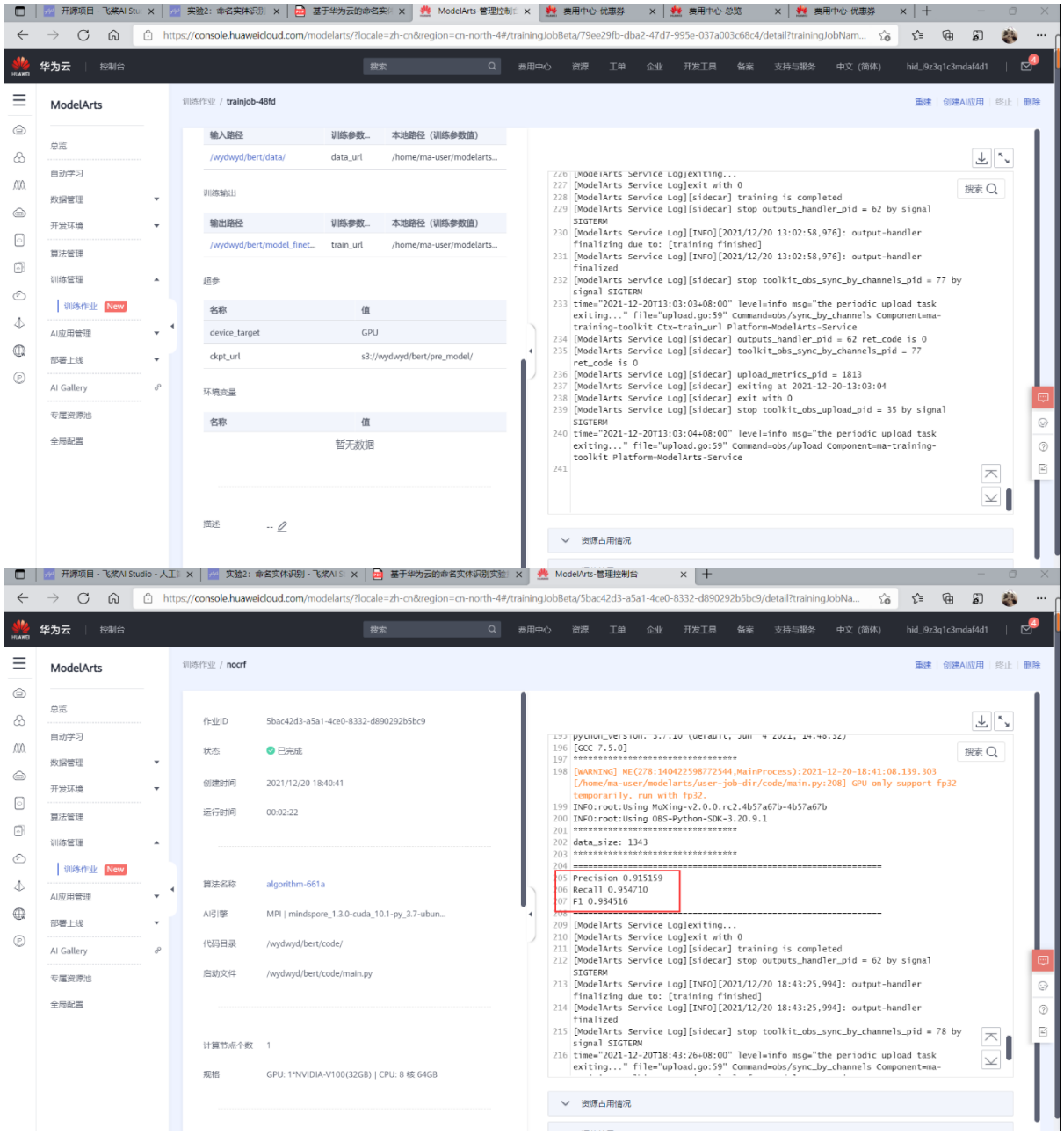
哈尔滨工业大学　　2021 学年秋季学期　　自然语言处理



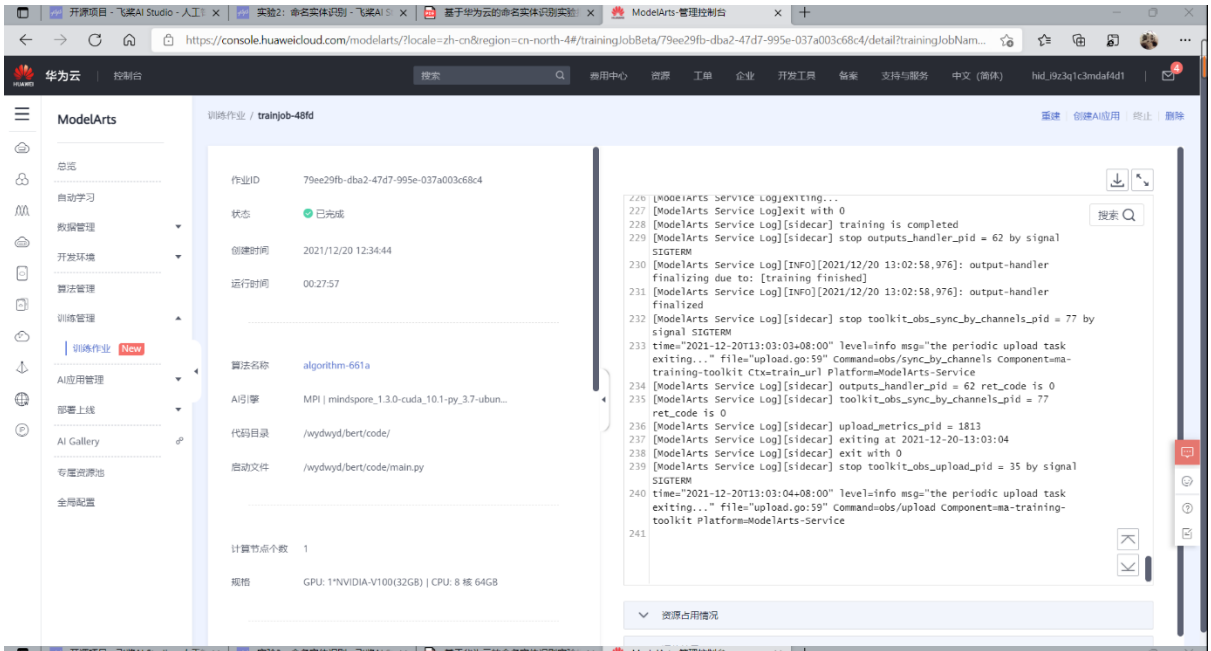


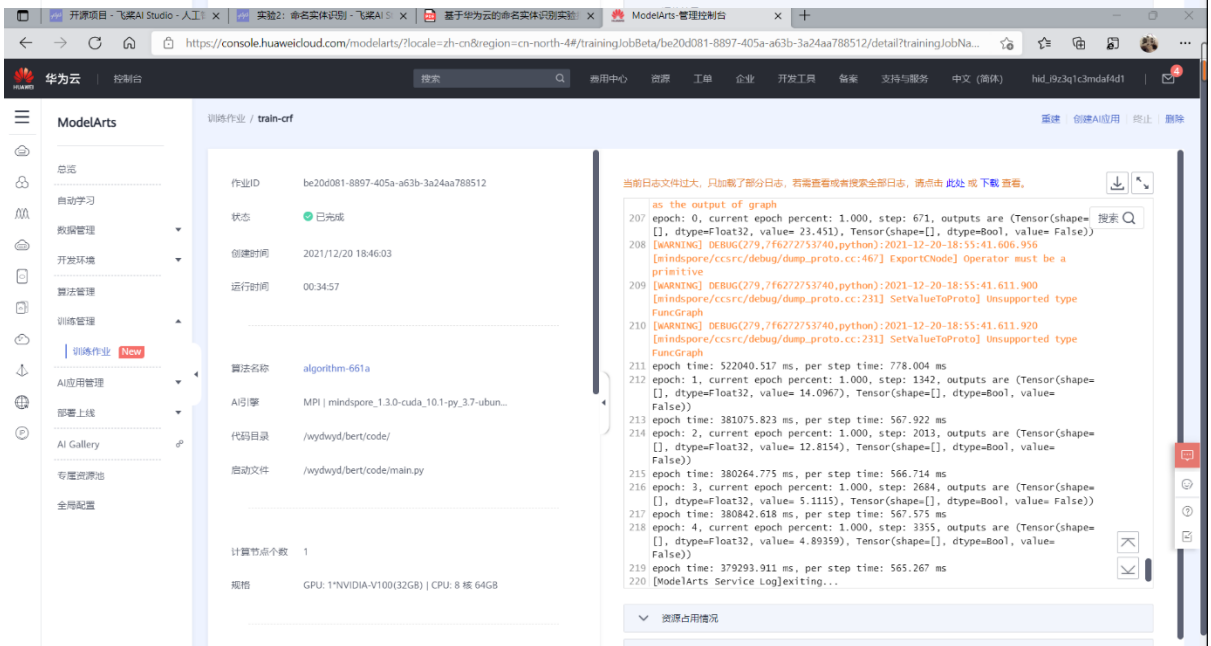


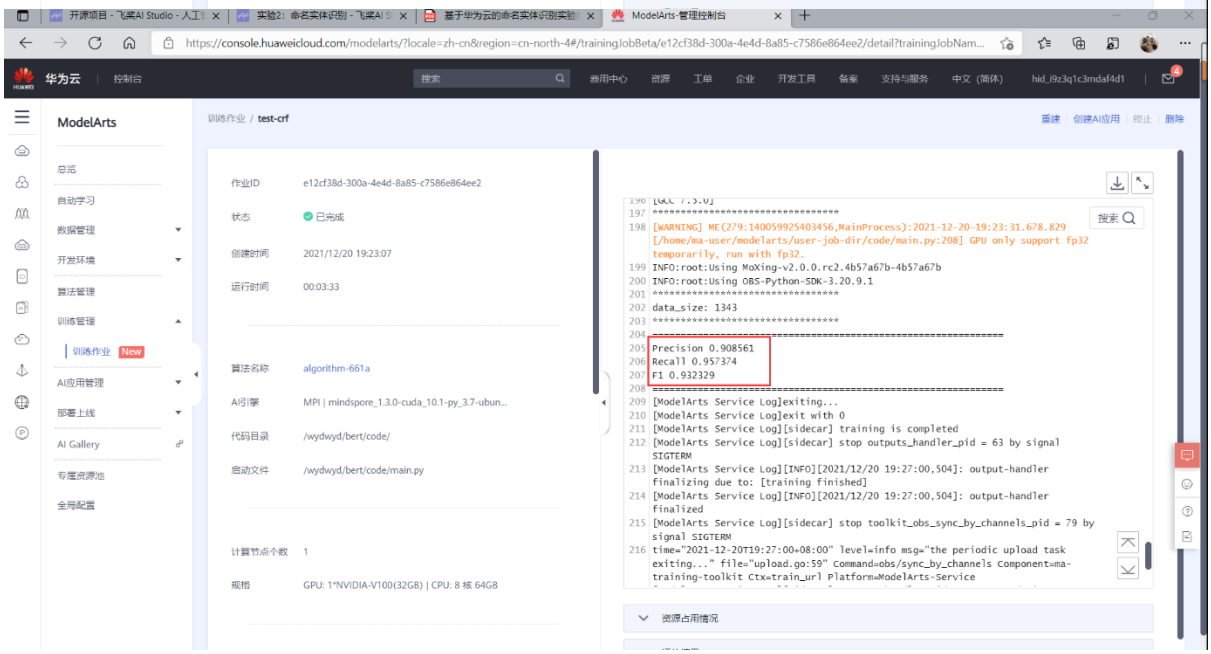
哈尔滨工业大学　　2021 学年秋季学期　　自然语言处理

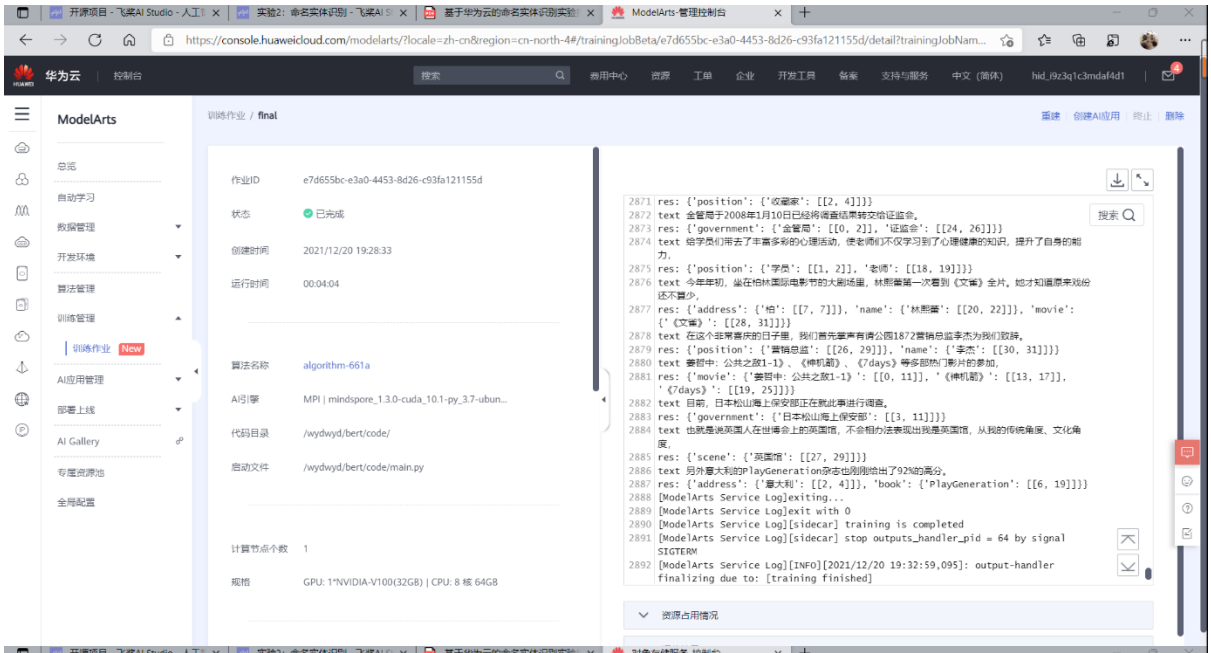


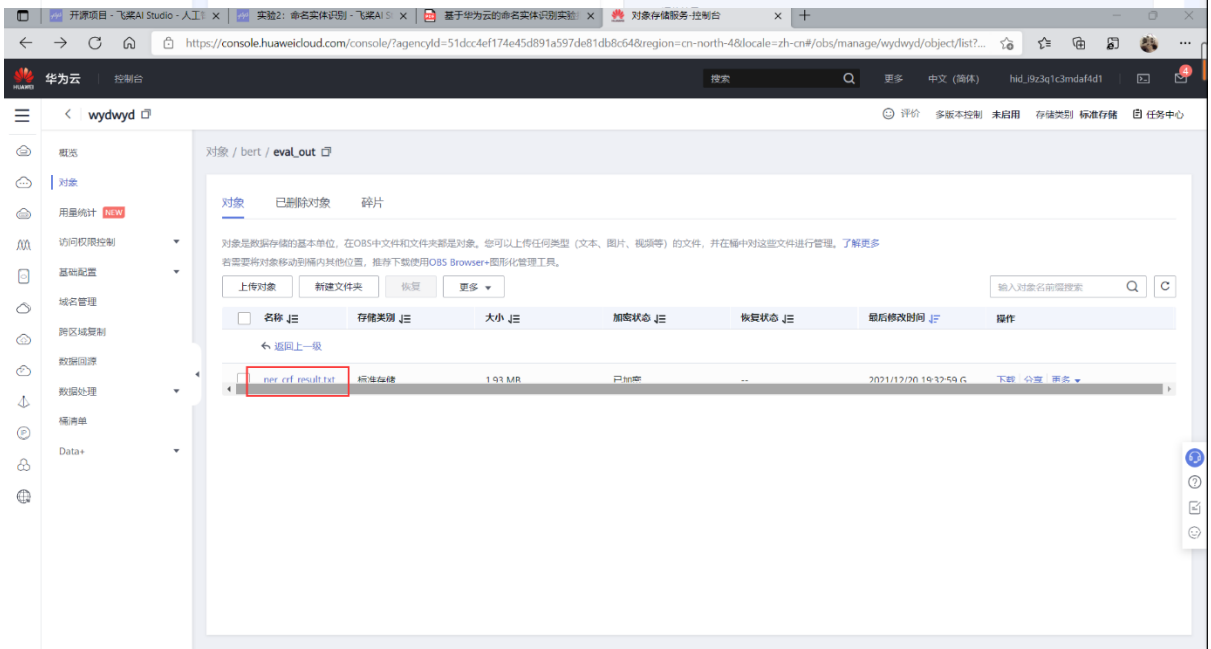
哈尔滨工业大学　　2021 学年秋季学期　　自然语言处理

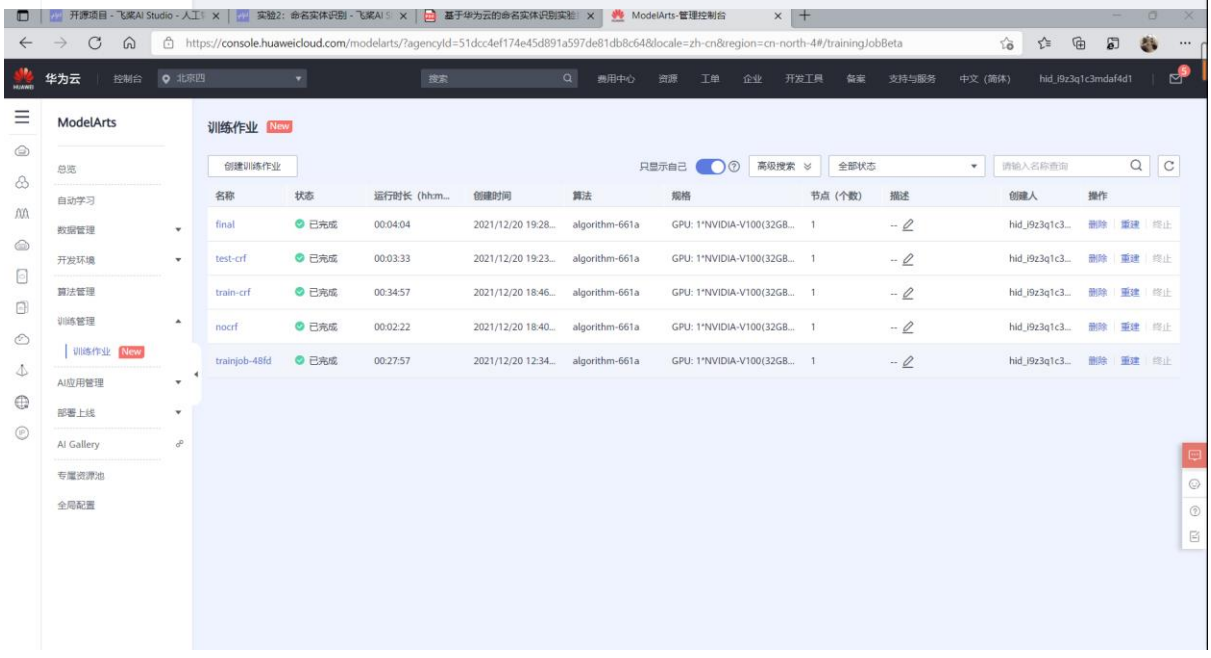







ModelArts
训练作业 / final
作业ID e7d655bc-e3a0-4453-8d26-c93fa121155d
状态 已完成
创建时间 2021/12/20 19:28:33
运行时间 00:04:04
算法名称 algorithm-661a
AI引擎 MPI | mindspore_1.3.0-cuda_10.1-py_3.7-ubun...
代码目录 /wydwyd/bert/code/
启动文件 /wydwyd/bert/code/main.py
计算节点个数 1
规格 GPU: 1*NVIDIA-V100(32GB) | CPU: 8 核 64GB
2871 res: {'position': {'收藏家': [[2, 4]]}}
2872 text 金管局于2008年1月10日已经将诸葛信票转交给证监会。
2873 res: {'government': {'金管局': [[0, 2]], '证监会': [[24, 26]]}}
2874 text 给学员们带去了丰富多彩的心理活动，使老师们不仅学习到了心理健康的知识，提升了自身的能力，
2875 res: {'position': {'学员': [[1, 2]], '老师': [[18, 19]]}}
2876 text 今年年初，坐在柏林国际电影节的大剧场里，林熙蕾第一次看到《文雀》全片，她才知道原来戏份还不算少，
2877 res: {'address': {'柏': [[7, 7]]}, 'name': {'林熙蕾': [[20, 22]]}, 'movie': {'《文雀》': [[28, 31]]}}
2878 text 在这个非常喜庆的日子里，我们首先荣声有请公园1872营销总监李杰为我们致辞。
2879 res: {'position': {'营销总监': [[26, 29]]}, 'name': {'李杰': [[30, 31]]}}
2880 text 要暂中：公共之敌1-1》、《神机箭》、《7days》等多部热门影片的参加，
2881 res: {'movie': {'要暂中：公共之敌1-1》': [[0, 11]], '《神机箭》': [[13, 17]], '《7days》': [[19, 25]]}}
2882 text 目前，日本松山海上保安部正在就此事进行调查。
2883 res: {'government': {'日本松山海上保安部': [[3, 11]]}}
2884 text 也就是说英国人在世博会上的英国馆，不会相办法表现出我是英国馆，从我的传统角度、文化角度，
2885 res: {'scene': {'英国馆': [[27, 29]]}}
2886 text 另外意大利的PlayGeneration杂志也刚刚给出了92%的高分。
2887 res: {'address': {'意大利': [[2, 4]]}, 'book': {'PlayGeneration': [[6, 19]]}}
2888 [ModelArts Service Log]exiting...
2889 [ModelArts Service Log]exit with 0
2890 [ModelArts Service Log][sidecar] training is completed
2891 [ModelArts Service Log][sidecar] stop outputs_handler_pid = 64 by signal SIGTERM
2892 [ModelArts Service Log][INFO][2021/12/20 19:32:59,095]: output-handler finalizing due to: [training finished]
资源占用情况


wydwyd
对象 / bert / eval_out
对象
已删除对象
碎片
对象是数据存储的基本单位，在OBS中文件和文件夹都是对象。您可以上传任何类型（文本、图片、视频等）的文件，并在桶中对这些文件进行管理。了解更多
若需要将对象移动到桶内其他位置，推荐下载使用OBS Browser+图形化管理工具。
上传对象
新建文件夹
恢复
更多
名称
存储类别
大小
加密状态
恢复状态
最后修改时间
操作
返回上一级
ner_crf_result.txt
标准存储
1.93 MB
已加密
2021/12/20 19:32:59 G
下载 分享 更多


ModelArts
训练作业 New
创建训练作业
只显示自己
高级搜索
全部状态
名称
状态
运行时长 (hh:m...
创建时间
算法
规格
节点（个数）
描述
创建人
操作
final 已完成 00:04:04 2021/12/20 19:28... algorithm-661a GPU: 1*NVIDIA-V100(32GB... 1 -- hid_i9z3q1c3... 删除 重建 终止
test-crf 已完成 00:03:33 2021/12/20 19:23... algorithm-661a GPU: 1*NVIDIA-V100(32GB... 1 -- hid_i9z3q1c3... 删除 重建 终止
train-crf 已完成 00:34:57 2021/12/20 18:46... algorithm-661a GPU: 1*NVIDIA-V100(32GB... 1 -- hid_i9z3q1c3... 删除 重建 终止
nocrf 已完成 00:02:22 2021/12/20 18:40... algorithm-661a GPU: 1*NVIDIA-V100(32GB... 1 -- hid_i9z3q1c3... 删除 重建 终止
trainjob-48fd 已完成 00:27:57 2021/12/20 12:34... algorithm-661a GPU: 1*NVIDIA-V100(32GB... 1 -- hid_i9z3q1c3... 删除 重建 终止

7. 实验的收获和体会

- ✧ 本次实验使我加深了对 HMM,CRF,ME 三种模型的理解
- ✧ 从代码出发，从实践中切身体会三种模型的算法流程与差异
- ✧ 从两个角度词与实体级别研究了命名实体识别的评测方法
- ✧ 对 HMM，ME，CRF 三种序列标记模型的优缺点进行了比较

参考文献

[1] https://www.cnblogs.com/nxfrabbit75/archive/2019/04/18/10727769.html#auto-id-10
[2] 李航,《统计学习方法》（第三版）
[3] 王江伟. 基于最大熵模型的中文命名实体识别[D]. 南京理工大学, 2005.
[4] https://zhuanlan.zhihu.com/p/88690315