Project2 说明文档

姓名: 林奕铖

学号: 20302010008

Part1. HMM 实现 NER 任务

代码基本结构

文件目录结构

```
│
# 测试集及验证集数据
├─ Chinese # 中文数据集
├─ English # 英文数据集
# 存储的模型
├─ hmm_model
│   ├─ Chinese.pkl # HMM应用于中文数据集的模型
│   └─ English.pkl # HMM应用于英文数据集的模型
# 源代码
├─ Part1.py # 包括模型定义、训练及测试过程的源代码
├─ Common.py # 包括数据读入模块和结果输出模块
└─ check.py # 用于生成f1-score等分数报告
```

- **代码基本结构**
  - **`class HMM`**

    实现了HMM隐马尔可夫模型

    `__init__`：初始化函数，接收所有可能的状态和观测值作为输入，初始化状态转移概率、观测概率和初始状态概率。

    `train`：训练函数，接收一组状态序列和对应的观测序列，用于计算和更新模型的状态转移概率、观测概率和初始状态概率。

    `__viterbi`：私有函数，供给`test`函数调用，实现了Viterbi算法，用于预测给定观测序列对应的最可能的状态序列。

    `test`：测试函数，接收一组观测序列，返回对应的预测状态序列。
  - **`class ModelLoader`**

    提供接口用于方便地训练、加载并测试模型

    `load_from_train`：从训练集中加载模型。首先从训练集中读取数据，然后使用这些数据创建并训练一个HMM模型。

    `load_from_file`：直接从pickle文件中加载HMM模型。

    `save_model`：保存模型，将HMM模型保存为pickle文件。

    `test`：测试模型。首先从测试集中读取数据，然后使用HMM模型预测标签，并将预测结果写入指定的文件。

`check`：检查预测结果。调用`check.py`比较预测结果和标准答案，检查预测的准确性。

实验结果 (Validation)

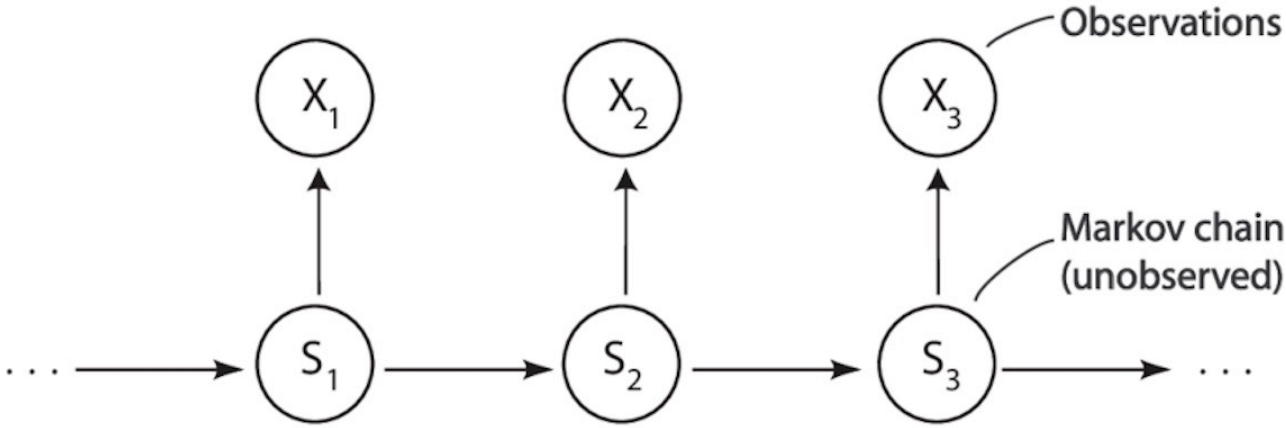
Part1-中文

	precision	recall	f1-score	support
B-NAME	0.9434	0.9804	0.9615	102
M-NAME	0.8675	0.9600	0.9114	75
E-NAME	0.8679	0.9020	0.8846	102
S-NAME	0.8000	0.5000	0.6154	8
B-CONT	0.8462	1.0000	0.9167	33
M-CONT	0.8889	1.0000	0.9412	64
E-CONT	0.9167	1.0000	0.9565	33
S-CONT	0.0000	0.0000	0.0000	0
B-EDU	0.8655	0.9717	0.9156	106
M-EDU	0.8551	1.0000	0.9219	177
E-EDU	0.8814	0.9811	0.9286	106
S-EDU	0.0000	0.0000	0.0000	0
B-TITLE	0.8305	0.8607	0.8453	689
M-TITLE	0.8362	0.8871	0.8609	1479
E-TITLE	0.9383	0.9710	0.9544	689
S-TITLE	0.0000	0.0000	0.0000	0
B-ORG	0.8966	0.9138	0.9051	522
M-ORG	0.9221	0.9081	0.9150	3622
E-ORG	0.7936	0.8103	0.8019	522
S-ORG	0.0000	0.0000	0.0000	0
B-RACE	0.6667	1.0000	0.8000	14
M-RACE	0.0000	0.0000	0.0000	0
E-RACE	0.6667	1.0000	0.8000	14
S-RACE	0.0000	0.0000	0.0000	1
B-PRO	0.3000	0.6667	0.4138	18
M-PRO	0.2500	0.6061	0.3540	33
E-PRO	0.4250	0.9444	0.5862	18
S-PRO	0.0000	0.0000	0.0000	0
B-LOC	0.2000	1.0000	0.3333	2
M-LOC	0.1667	1.0000	0.2857	6
E-LOC	0.2222	1.0000	0.3636	2
S-LOC	0.0000	0.0000	0.0000	0
micro avg	0.8696	0.9046	0.8868	8437
macro avg	0.5265	0.6832	0.5679	8437
weighted avg	0.8803	0.9046	0.8908	8437

Part1-英文

	precision	recall	f1-score	support
B-PER	0.9614	0.6900	0.8034	1842
I-PER	0.9358	0.7689	0.8442	1307
B-ORG	0.8110	0.7554	0.7822	1341
I-ORG	0.8600	0.6791	0.7589	751
B-LOC	0.9187	0.8247	0.8692	1837
I-LOC	0.8710	0.7354	0.7975	257
B-MISC	0.9257	0.8113	0.8647	922
I-MISC	0.8014	0.6763	0.7335	346
micro avg	0.9002	0.7538	0.8205	8603
macro avg	0.8856	0.7426	0.8067	8603
weighted avg	0.9032	0.7538	0.8201	8603

HMM 基本原理



1. 如图所示，HMM(隐马尔可夫模型)假设当前隐状态仅与前一时刻的隐状态有关，与其他时刻的隐状态无关、与观测结果无关。同时假设观测结果仅与当前时刻的隐状态有关，与其他因素无关。
2. HMM模型的参数有三个，分别是转移概率矩阵、发射概率矩阵与初始概率矩阵。分别表示隐状态之间的转移概率、隐状态到观测值的发射概率和初始时刻每个隐状态的概率。

3. 在NER任务中，将实体标签看成隐状态，文字（单词）看成观测结果。训练过程只需要简单地统计每句话的句首单词分布频率即可得到初始概率矩阵。同时，统计所有的(标签->文字)分布频率即可得到发射概率矩阵，统计所有的(前一标签->后一标签)分布频率即可得到转移概率矩阵。
4. HMM模型的预测过程是根据观测序列来预测对应的隐状态序列。预测过程中使用Viterbi算法，通过动态规划的方式找到观测序列对应的最可能的隐状态序列。

算法的流程如下：

1. 初始化：定义一个二维矩阵viterbi，其中每个元素viterbi[i][j]表示在时刻i处于状态j的最大概率，同时定义一个二维矩阵backpointer，其中每个元素backpointer[i][j]表示在时刻i处于状态j时，其前一时刻的最可能的状态。
2. 对于时刻0，计算初始状态概率和观测概率的乘积，并将结果存储在viterbi[0][j]中，同时将backpointer[0][j]设置为0。
3. 对于时刻t（t>0），对于每个可能的隐状态j，计算上一时刻的最大概率乘以转移概率和观测概率的乘积，即viterbi[t][j] = max(viterbi[t-1][k] * 转移概率[k][j] * 观测概率[j][t])，同时记录backpointer[t][j]为使得viterbi[t][j]最大的状态k。
4. 对于最后一个时刻T，从所有可能的隐状态中选择概率最大的状态作为最终预测的隐状态序列的最后一个状态，即最大概率状态index = argmax(viterbi[T])。
5. 通过回溯，对于每个时刻t，从backpointer[t+1][index]中找到对应的状态指针。从而得到最佳路径。

Part2. CRF实现NER任务

代码基本结构

文件目录结构

```
│
# 测试集及验证集数据
├─ Chinese # 中文数据集
├─ English # 英文数据集
# 存储的模型
├─ crf_model
│  ├─ Chinese.pkl # CRF应用于中文数据集的模型
│  └─ English.pkl # CRF应用于英文数据集的模型
# 源代码
├─ Part2.py # 包括模型定义、训练及测试过程的源代码
├─ Common.py # 包括数据读入模块和结果输出模块
└─ check.py # 用于生成f1-score等分数报告
```

- **代码基本结构**
  - **`class CRF`**

    实现了CRF条件随机场模型

`__get_features_from_word`：私有方法，供train函数调用，从给定的句子和索引中提取特征。这些特征包括当前词的各种属性（如是否全大写、是否是标题、是否是数字等），以及前一个词和后一个词的一些属性（如果存在的话）。如果当前词是句子的开始或结束，还会添加特殊的'BOS'或'EOS'特征。

`__init__`：初始化函数，创建一个`sklearn_crfsuite.CRF`对象。

`train`：训练函数，接收一组状态序列和对应的观测序列，用于训练CRF模型。首先从观测序列中提取特征，然后使用这些特征和对应的状态序列来训练模型。

`test`：测试函数，接收一组观测序列，返回对应的预测状态序列。首先从观测序列中提取特征，然后使用CRF模型来预测每个观测序列的状态。

▼ **`class ModelLoader`**

提供接口用于方便地训练、加载并测试模型

`load_from_train`：从训练集中加载模型。首先从训练集中读取数据，然后使用这些数据创建并训练一个CRF模型。

`load_from_file`：直接从pickle文件中加载CRF模型。

`save_model`：保存模型，将CRF模型保存为pickle文件。

`test`：测试模型。首先从测试集中读取数据，然后使用CRF模型预测标签，并将预测结果写入指定的文件。

`check`：检查预测结果。调用`check.py`比较预测结果和标准答案，检查预测的准确性。

实验结果 (Validation)

Part2-中文

	precision	recall	f1-score	support
B-NAME	1.0000	0.9804	0.9901	102
M-NAME	1.0000	0.9733	0.9865	75
E-NAME	1.0000	0.9804	0.9901	102
S-NAME	1.0000	1.0000	1.0000	8
B-CONT	1.0000	1.0000	1.0000	33
M-CONT	1.0000	1.0000	1.0000	64
E-CONT	1.0000	1.0000	1.0000	33
S-CONT	0.0000	0.0000	0.0000	0
B-EDU	0.9905	0.9811	0.9858	106
M-EDU	0.9888	1.0000	0.9944	177
E-EDU	0.9905	0.9811	0.9858	106
S-EDU	0.0000	0.0000	0.0000	0
B-TITLE	0.9017	0.9057	0.9037	689
M-TITLE	0.8809	0.9053	0.8930	1479
E-TITLE	0.9769	0.9826	0.9797	689
S-TITLE	0.0000	0.0000	0.0000	0
B-ORG	0.9553	0.9406	0.9479	522
M-ORG	0.9373	0.9613	0.9492	3622
E-ORG	0.8891	0.8755	0.8822	522
S-ORG	0.0000	0.0000	0.0000	0
B-RACE	1.0000	1.0000	1.0000	14
M-RACE	0.0000	0.0000	0.0000	0
E-RACE	1.0000	1.0000	1.0000	14
S-RACE	0.0000	0.0000	0.0000	1
B-PRO	0.8889	0.8889	0.8889	18
M-PRO	0.8889	0.7273	0.8000	33
E-PRO	0.8889	0.8889	0.8889	18
S-PRO	0.0000	0.0000	0.0000	0
B-LOC	1.0000	1.0000	1.0000	2
M-LOC	1.0000	1.0000	1.0000	6
E-LOC	1.0000	1.0000	1.0000	2
S-LOC	0.0000	0.0000	0.0000	0
micro avg	0.9311	0.9435	0.9372	8437
macro avg	0.7243	0.7179	0.7208	8437
weighted avg	0.9312	0.9435	0.9372	8437

Part2-英文

	precision	recall	f1-score	support
B-PER	0.8977	0.8811	0.8893	1842
I-PER	0.9495	0.9487	0.9491	1307
B-ORG	0.8651	0.8084	0.8358	1341
I-ORG	0.8213	0.8202	0.8208	751
B-LOC	0.9107	0.9053	0.9080	1837
I-LOC	0.9181	0.8288	0.8712	257
B-MISC	0.9431	0.8265	0.8809	922
I-MISC	0.8780	0.7283	0.7962	346
micro avg	0.9012	0.8663	0.8834	8603
macro avg	0.8979	0.8434	0.8689	8603
weighted avg	0.9013	0.8663	0.8829	8603

CRF基本原理

线性链条件随机场是特殊的马尔科夫随机场，满足马尔科夫性:
$P(Y_i \mid X, Y_1, \cdots, Y_{i-1}, Y_{i+1}, \cdots, Y_n) = P(Y_i \mid X, Y_{i-1}, Y_{i+1})$

在给定观测序列x的条件下，路径y的分数可表示为$P(y \mid x) = \frac{1}{Z(x)} \exp \sum_{k=1}^{K} w_k f_k(y, x)$.

其中w_k为权重，需要通过模型训练调整。 f_k为特征函数，取值为0或1. $Z(x)$为规范化因子。

在NER任务中，一些常用的特征函数包括当前词的各种属性（如是否全大写、是否是标题、是否是数字等），以及前一个词和后一个词的一些属性（如果存在的话）。如果当前词是句子的开始或结束，还会添加特殊的'BOS'或'EOS'特征。

训练过程中，首先从给定的观测序列中提取特征，并将提取的特征和对应的状态序列作为输入，通过最大化似然函数的方法来调整模型的权重w_k，使得模型能够更好地拟合训练数据。

模型的预测过程依然使用Viterbi算法，通过动态规划的方式找到观测序列对应的最可能的隐状态序列。

Part3. BiLSTM+CRF实现NER任务

代码基本结构

文件目录结构

```
│
# 测试集及验证集数据
├─ Chinese # 中文数据集
├─ English # 英文数据集
# 存储的模型
├─ bi_model
│  ├─ Chinese.pkl # BiLSTM_CRF应用于中文数据集的模型
│  └─ English.pkl # BiLSTM_CRF应用于英文数据集的模型
# 源代码
├─ Part3.py # 包括模型定义、训练及测试过程的源代码
├─ Common.py # 包括数据读入模块和结果输出模块
└─ check.py # 用于生成f1-score等分数报告
```

- **代码基本结构**
  - **`class BiLSTM_CRF`**

    实现了BiLSTM_CRF模型

    `__init__`：初始化函数，设置了词嵌入层、LSTM层、全连接层和转移矩阵等参数。

    `init_hidden`：初始化隐藏状态的函数。

    `_get_lstm_features`：将输入的句子通过词嵌入和BiLSTM层，得到每个词的隐藏状态。

    `_forward_alg`：实现了前向算法，计算所有可能的路径的分数和。

    `_score_sentence`：计算给定路径的分数。

    `_viterbi_decode`：实现了维特比算法，找到最可能的标签序列。

    `neg_log_likelihood`：计算负对数似然值，用于训练模型。

`forward`：利用训练好的模型预测给定句子的标签。

- **`class ModelLoader`**

  提供接口用于方便地训练、加载并测试模型

  `load_from_train`：从训练集中加载模型。首先从训练集中读取数据，然后使用这些数据创建并训练一个CRF模型。

  `load_from_file`：直接从pickle文件中加载CRF模型。

  `save_model`：保存模型，将CRF模型保存为pickle文件。

  `test`：测试模型。首先从测试集中读取数据，然后使用CRF模型预测标签，并将预测结果写入指定的文件。

  `check`：检查预测结果。调用`check.py`比较预测结果和标准答案，检查预测的准确性。

实验结果 (Validation)

Part3-中文

	precision	recall	f1-score	support
B-NAME	0.9126	0.9216	0.9171	102
M-NAME	0.8500	0.9067	0.8774	75
E-NAME	0.9423	0.9608	0.9515	102
S-NAME	0.7500	0.7500	0.7500	8
B-CONT	1.0000	0.9697	0.9846	33
M-CONT	1.0000	0.9688	0.9841	64
E-CONT	1.0000	0.9697	0.9846	33
S-CONT	0.0000	0.0000	0.0000	0
B-EDU	0.9717	0.9717	0.9717	106
M-EDU	0.9831	0.9887	0.9859	177
E-EDU	0.9811	0.9811	0.9811	106
S-EDU	0.0000	0.0000	0.0000	0
B-TITLE	0.9060	0.8955	0.9007	689
M-TITLE	0.9053	0.9053	0.9053	1479
E-TITLE	0.9854	0.9826	0.9840	689
S-TITLE	0.0000	0.0000	0.0000	0
B-ORG	0.9638	0.9176	0.9401	522
M-ORG	0.9426	0.9572	0.9499	3622
E-ORG	0.8846	0.8812	0.8829	522
S-ORG	0.0000	0.0000	0.0000	0
B-RACE	1.0000	1.0000	1.0000	14
M-RACE	0.0000	0.0000	0.0000	0
E-RACE	1.0000	1.0000	1.0000	14
S-RACE	0.0000	0.0000	0.0000	1
B-PRO	0.8889	0.8889	0.8889	18
M-PRO	0.8205	0.9697	0.8889	33
E-PRO	0.9474	1.0000	0.9730	18
S-PRO	0.0000	0.0000	0.0000	0
B-LOC	0.6667	1.0000	0.8000	2
M-LOC	1.0000	1.0000	1.0000	6
E-LOC	0.6667	1.0000	0.8000	2
S-LOC	0.0000	0.0000	0.0000	0
micro avg	0.9348	0.9384	0.9366	8437
macro avg	0.6865	0.7121	0.6969	8437
weighted avg	0.9349	0.9384	0.9365	8437

Part3-英文

	precision	recall	f1-score	support
B-PER	0.9215	0.7389	0.8201	1842
I-PER	0.9465	0.7307	0.8247	1307
B-ORG	0.8787	0.7241	0.7939	1341
I-ORG	0.8842	0.7017	0.7825	751
B-LOC	0.9474	0.8144	0.8759	1837
I-LOC	0.9431	0.7743	0.8504	257
B-MISC	0.9033	0.7907	0.8433	922
I-MISC	0.8992	0.6445	0.7508	346
micro avg	0.9188	0.7510	0.8265	8603
macro avg	0.9155	0.7399	0.8177	8603
weighted avg	0.9187	0.7510	0.8260	8603

BiLSTM基本原理

在第二部分的CRF中需要自行设计特征函数，而LSTM可以直接提取特征，不用再自行设计特征模板。BiLSTM计算出每一个词汇的隐状态发射概率，将其传输给CRF部分。

BiLSTM（Bidirectional Long Short-Term Memory）双向长短时记忆网络是一种循环神经网络（RNN）的变体。与传统的RNN只有一个方向的信息流动不同，BiLSTM允许信息的双向流动，即同时考虑当前时刻的前向和后向上下文信息。BiLSTM由两个LSTM层组成，一个负责前向信息流动，另一个负责后向信息流动。每个LSTM层由多个LSTM单元组成，每个LSTM单元可以根据当前输入和前一个时刻的隐藏状态，更新当前时刻的隐藏状态和记忆状态。通过这种方式，BiLSTM能够有效地捕捉到上下文信息，提取更丰富的特征。

$$score(x, y) = \sum_{i=0}^{n} transitions_{y_{i+1}y_i} + \sum_{i=1}^{n} feats_{i,y_i}$$

$$P(y \mid x) = \frac{exp(score(x, y))}{\sum_{all\ possible\ \tilde{y}} exp(score(x, \tilde{y}))}$$

$$-logP(y \mid x) = -log\frac{exp(score(x, y))}{\sum_{all\ possible\ \tilde{y}} exp(score(x, \tilde{y}))}$$

$$= -score(x, y) + log \sum_{all\ possible\ \tilde{y}} exp(score(x, \tilde{y})$$