自然语言处理课程 中文医学语料分詞作业报告

北京大学软件与微电子学院　　程咏　詹承勋

自然语言处理 (Natural Language Processing, NLP) 近年来在人工智能领域为较热门的研究方向，尤其是在中文语料的应用领域还有着很大的发展空间。在 NLP 的研究上，文本分词与词性标注 (Part Of Speech, POS)、命名实体识别 (Named Entity Recognition, NER) 为最基本但也是最重要的第一步，分词与词性标注的好坏，往往影响着往后的分析与研究成果。常用于分词的模型与算法有着机器学习模型的 HMM[1]，判别式模型的 CRF[2, 3]，SVM 等，深度学习模型[4]有 RNN，LSTM[5]，LSTM+CRF[6] 等。在目前各种模型中词性标注的准确率往往可以达到 84%~94% 间的准确率[7]，而整句则可达到约 56% 的确率，命名实体则为 81.01% 至 92.33% 间[8]。

在以基于词典的分词与标注中，生成式模型分词算法的 n-gram 与 隐马尔可夫 (Hidden Markov Model, HMM) 模型应用较多。举例而言，广受欢迎的开源分词与词性标注工具，例如使用 Python 语言的 jieba[9] 与基于 Java 语言开发的 HanLP[10] 都使用 HMM 模型进行实作。是故考量到实作难度以及模型修改的可靠性，本报告分词、词性标注与命名实体识别三部分皆选用较成熟的 HMM + Viterbi 算法作为主要实作的标注模型算法，并逐步地进行参数与模型架构的调整，与完成最终的测试与模型评价。在最后的结果我们使用已分词的训练集可以得到 POS 的准确率为 93 ~ 98 %，而 NER 则为 80 ~ 84 %，自行分词的训练集得到 POS 为 69 ~ 72 %，NER 为 45 ~ 46 %。

一、实验目的:

本报告的目的为设计一个基于 HMM 与 Viterbi 算法的词性标注与命名实体识别模型，在给予的医学相关有限训练语料中，进行测试集语料的词性标注与命名实体识别，并针对资料前处理以及各项模型参数对模型进行调优与完善，进而对各版本迭代模型做出评估与比较。

二、实验原理:

马尔可夫链 (Markov Chain, MC) 叙述了状态分布矩阵 Sn 与转移概率矩阵 P 间的依据随机变量随时间按照马尔可夫性质进行变化的过程。
而在隐马尔可夫模型中，建立了一串可见状态链与隐含状态链，并于隐含状态链中存在转换概率 (transition probability)，而这马尔科夫链中服从马尔可夫性

质的”无记忆性”，也就是当前状态只受前一状态的影响，而不受更之前状态的影响。因此可以从可观察的参数中确定该过程的隐含参数，于 NLP 词性标注中，则是藉由极大或然估计，通过训练集的词性计数来学习 HMM 相关的参数，包含初始状态概率、转移概率、以及发射概率的学习，如下**图一**所示。

初始状态概率的学习：$\pi[q] = \frac{\text{词性}q\text{出现在所有训练句子开头的次数}}{\text{总训练句子数}}$

转移概率的学习：

$transition[q_1][q_2] = \frac{\text{所有训练句子的所有相邻词性二元组中，}q_1\text{词性后面跟着}q2\text{的次数}}{\text{所有训练句子的所有相邻词性二元组中，第一个词性是}q_1\text{的二元组个数}}$

发射概率的学习：

$emission[q][w] = \frac{\text{所有句子中所有的词性－词二元组中，词性}q\text{发射词}w\text{的次数}}{\text{所有句子中所有的词性－词二元组中，词性是}q\text{的个数}}$

图一、HMM 模型概率的学习[11]

HMM 模型一般为前向算法 (forward algorithm)、维特比算法 (Viterbi algorithm)、前向-后向算法 (forward-backward algorithm) 所组成，分别为进行匹配观察序列 (observation sequence) 最匹配可能的系统、确定观察序列最可能的隐藏状态序列 (state sequence)、决定已生成的观察序列最可能的模型参数。本报告是依据输入的已标注训练集文本语句输出其对应的词性序列，因此语句中的单词即为观察符号，词性标注序列则为隐藏状态。模型本身λ则使用初始概率分布π、转移概率分布 A、观测(发射)概率分布 B，即隐马尔可夫三要素来确定λ = (A, B, π)[12]。在 HMM 词性标注中，转移概率 transition[i][j] 是词性 i 条件下下一个词性是 j 的概率，即 P(j | i)；发射概率 emission[i][w] 是词性 i 产生词 w 的概率，即 P(w | i)[11]。

因此藉由训练完 HMM 模型，产生三个概率矩阵后，我们可以判断单词的上下文或然率进行词性标注。维特比算法为求解 HMM 上的最短路径（-log(prob)，也即是最大概率）的算法。使用维特比算法可以用来搜索已知观察序列与 HMM 模型下最可能的隐藏状态序列，也就是单词最可能的词性标注。而在进行词性标注时，一开始必须选定语料库的训练标记集，于此次报告中，作业要求中所提供的各项医学相关语料库做为我组的标记集。基于 HMM 的标记算法为使用该训练语料库作为随机标注的算法基础，以给定之训练语料库计算测试集上下文某一单词的词性标记机率并对文本进行标注。

三、实验内容:

程序详细介绍:

此次程序主要执行流程模块如下**图二**所示。

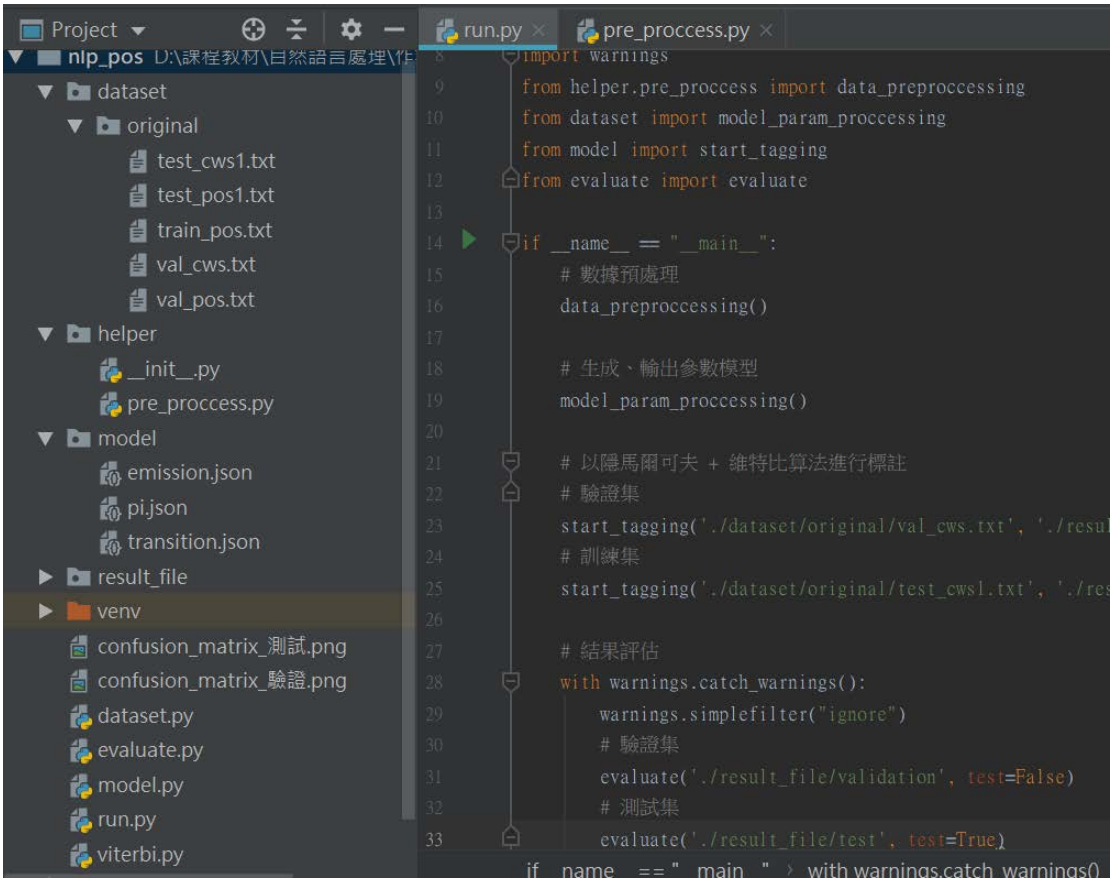
图二、实验步骤流程设计

以 POS 部分为例，此次报告程序设计结构分作如下**表一**所示，分词与 NER 部分为相同架构。

档名	**功能**
(1). **run.py**	主程序，调用以下文件中函数或类别，完成所有训练、测试、预测过程
(2). **pre_proccess.py**	数据预处理，载入测试集test_pos1.txt 以及验证集 val_pos.txt
(3). **dataset.py**	载入训练集数 train_pos.txt、 生成、输出参数模型 [模型输出包含 emission.json、pi.json、tagged.json、target.json、transition.json]
(4). **model.py**	载入参数模型并读取测试集后引用 viterbi.py 进行标注，并输出标注结果
(5). **viterbi.py**	维特比标注算法
(6). **evaluate.py**	引入前项处理好的测试集及验证集数据，进行比对并计算准确率
(7) **dataset/original folder**	原始数据集，包含训练/测试/验证集
(8) modle/**emission.json、pi.json、transition.json**	储存训练初始/转移/发射概率的模型
(9) result_file/test/**tagged.json、target.json** result_file/validatation/**tagged.json、target.json**	储存测试集/验证集 的已预处理字词-词性对数据、已标注字词-词性对数据

表一、程序设计结构与功能

代码结构如下**图三**所示。



图三、POS 程序分层设计

(1) run.py

主程序分别呼叫执行数据预处理、生成 / 输出参数模型、标注验证集与测试集、验证集与测试集结果评估。

(2) pre_proccess.py

载入测试集手动标注范例语料 test_pos1.txt，并将其中已标注词性字段依据空格与词性标注自符 "/"，分切后输出成 json 档案 result_file/test/target.json 留待之后与本次报告程式标注测试集评估准确率。这边将字词与词性分切转化成 json 档 [字词, 词性] 的 List 是为了往后比对方便。验证集同理，载入 val_pos.txt，输出 result_file/validation/target.json。

数据预处理部分为使用训练语料中的单字字词与其已标注的词性，将其依据空格与换行符号等分切处理产生一个**字词列表**与**词性列表**，并依据字词列表中各词性出现次数计算并产生一个词性初始概率字典**(词频字典)**，另可以依据设定的最小保留词的频率参数，筛选掉出现次数过低的词性。最后再使用词性列表产生一个不重复的**词性集合**。另在 NER 中训练集方括号多重嵌套问题也使用以下代码 **(图四)** 做数据清洗方便对其做出处理，详细原理于**五、误差分析** 與 **六、思考题**中介绍。

```python
def train_handle(inputfile, outputfile, output_path):
    if not os.path.exists(output_path):
        os.makedirs(output_path)
    with open(inputfile,'r') as inp, open('./'+output_path+'/'+outputfile+'.txt','w', encoding='utf-8-sig') as outp:
        for string in inp.readlines():
            i=0
            substring = ""
            tag = ""
            temp = []
            res = []
            stack = []

            while i<len(string):

                if string[i] != '[' and string[i] != ']' and string[i] != ' ':
                    substring = substring + string[i]
                    if i==(len(string)-1):
                        if substring != '':
                            stack.append(substring)
                            while len(stack)>0:
                                temp.append(stack.pop()+'/0')
                            res.append(temp[::-1])
                            temp=[]

                elif string[i] == ' ':
                    if substring != "":
                        stack.append(substring)
                        substring = ""

                elif string[i] == '[':
                    isInstack = '[' in stack

                    if (isInstack) or len(stack) == 0:
                        stack.append('[')

                    elif len(stack)>0:
                        while len(stack)>0:
                            temp.append(stack.pop()+'/0')

                        res.append(temp[::-1])

                        temp=[]
                        stack.append('[')

                elif string[i] == ']':
                    if substring != "":
                        stack.append(substring)
                        substring = ""
                    if string[i+1] == 'n':
                        tag = string[i+1]+string[i+2]
                        i+=2
                    else:
                        tag = string[i+1]+string[i+2]+string[i+3]
                        i+=3
                    while len(stack)>0:
                        pop = stack.pop()

                        if pop != '[':
                            temp.append(pop+"/"+tag)

                        else:

                            isInstack = ('[' in stack)
                            if (isInstack):
                                for q in temp[::-1]:
                                    stack.append(q)
                                temp=[]

                            break
                    if len(temp) != 0:
                        res.append(temp[::-1])

                    temp=[]
                    tag=""
                i+=1

            for sent in res:
                for word in sent:
                    outp.write(word+' ')
            outp.write("\n")
```

图四、NER 数据清洗核心代码

(3) dataset.py

此部分载入已标注之训练数据语料 train_pos.txt，并依据 HMM 算法训练产生三个输出字典概率模型，计算公式如上**图一**所示，包含初始概率(pi.json)、转移概率 (transition.json)、放射概率模型 (emission.json)，并根据最小词频过滤词汇表。

模型训练部分共输出三个概率模型，分别是初始概率、转移概率、放射概率模型。

(3.1) 使用上述提及的字词列表统计各个字词在整体训练集中出现的次数，并计算出其出现的概率，输出成**初始概率模型** (pi.json)。

(3.2) 使用不重复的词性集合、词性列表，算出相邻词的上下文出现相关概率，并输出**词性转移概率模型** (transition.json) 。

(3.3) 使用词性集合、词性列表、字词列表、字词集合去算出词性-词性间转换关系。并输出**放射概率模型** (emission.json)。

(4) model.py

载入了上述三个概率模型，并读取测试与验证集语料 test_cws1.txt/val_cws.txt 后载入维特比算法封装 (viterbi.py)**(图五)** 对寻找最佳路径 **(图六)** 其进行标注，标注完后输出标注文档 (tagged.json) 作为结果，并同时还原并输出 txt 文档使其与输入文档格式一致**(图七)**。

```python
def viterbi(obs, pi, transition, emission):
    viterbi_matrix = defaultdict(dict)
    traceback_matrix = defaultdict(dict)

    STATUS_LEN = len(transition)
    OBS_LEN = len(obs)

    # hidden states
    # 給出句子，計算其最有可能的 hidden state(詞性) 序列
    states = transition.keys()

    # 計算第一個詞的詞性，初始化兩個矩陣
    for state in states:
        # like normal in emission[healthy]
        if obs[0] in emission[state]:
            viterbi_matrix[0][state] = pi[state] * emission[state][obs[0]]
        else:
            viterbi_matrix[0][state] = 1e-4
        traceback_matrix[0][state] = '<start>'

    def get_max_from_dict(time, to_state):
        max_prob = max_state = 0
        for s in states:
            # example:
            # health -> health, health -> cold; health -> fever, fever -> cold
            if obs[time] in emission[to_state]:
                prob = viterbi_matrix[time-1][s] * transition[s][to_state] * emission[to_state][obs[time]]
            else:
                prob = viterbi_matrix[time-1][s] * transition[s][to_state] * 1e-4

            if prob > max_prob:
                max_prob = prob
                max_state = s

        return max_prob, max_state

    # 遞推，從第二個詞開始
    for time in range(1, OBS_LEN):  # 觀測序列長度
        for state in states:
            max_prob, max_state = get_max_from_dict(time, state)
            viterbi_matrix[time][state] = max_prob
            traceback_matrix[time][state] = max_state

    # 查找最後時刻的最大概率和相應狀態
    max_prob_final_state, max_prob = find_last_time_max_prob_and_state(states, viterbi_matrix, OBS_LEN)


    return traceback(traceback_matrix, max_prob_final_state, OBS_LEN)
```

图五、维特比算法核心代码

```python
def traceback(traceback_matrix, max_prob_final_state, T):
    best_path = [max_prob_final_state]
    for time in range(T-1, 0, -1):
        try:
            prev_state = traceback_matrix[time][best_path[-1]]
            best_path.append(prev_state)
        except:
            best_path.append(None)
    return list(reversed(best_path))
```

图六、回朔路径核心代码。

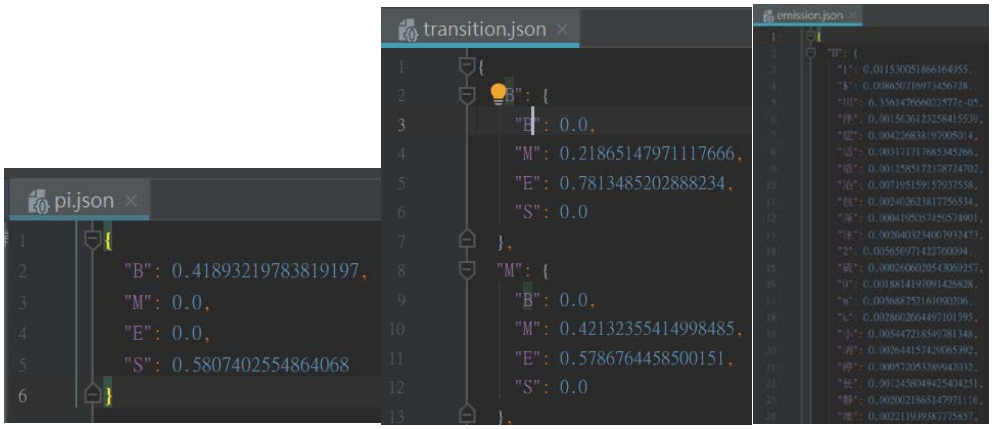
```python
def recover2file(read_list, ner_list, outputname, outputpath):
    '''
    將標注好的命名實體按照原格式輸出
    :param read_list: 詞列表
    :param ner_list: 命名實體列表
    :param outputname: 輸出文件名
    :param outputpath: 輸出目的地文件夾
    :return:
    '''
    # 追加寫回文本中
    with open(outputpath + '/' + outputname + '.txt', 'a', encoding='utf-8-sig') as outp:
        for line in range(len(read_list)):
            for word in range(len(read_list[line])):
                ner_split = ner_list[line][word].split('/')

                if word == len(read_list[line]) - 1:

                    if ner_list[line][word] == '0':
                        outp.write(read_list[line][word] + ' ')
                        break

                    elif len(ner_split) == 1:
                        S = ner_list[line][word][0]
                        NER = ner_list[line][word][2:]

                        if S == "B":
                            outp.write("[" + read_list[line][word] + "]" + NER + ' ')
                        else:
                            outp.write(read_list[line][word] + "]" + NER + ' ')

                    elif len(ner_split) == 2:
                        S1 = ner_split[0][0]
                        S2 = ner_split[1][0]
                        NER1 = ner_split[0][2:]
                        NER2 = ner_split[1][2:]

                        if S1 == 'B':
                            outp.write("[" + read_list[line][word] + "]" + NER1 + ']' + NER2 + ' ')
                        else:
                            outp.write(read_list[line][word] + "]" + NER1 + ']' + NER2 + ' ')

                elif ner_list[line][word] == '0':
                    outp.write(read_list[line][word] + ' ')

                elif len(ner_split) == 1:
                    S = ner_list[line][word][0]
                    NER = ner_list[line][word][2:]

                    if S == "B":
                        ner_split_next = ner_list[line][word + 1].split('/')

                        if len(ner_split_next) == 1:

                            if (ner_list[line][word + 1][0] == 'M' or ner_list[line][word + 1][0] == "E"):
                                outp.write("[" + read_list[line][word] + ' ')
                            else:
                                outp.write("[" + read_list[line][word] + "]" + NER + ' ')

                        elif len(ner_split_next) == 2:
                            if (ner_split_next[1][0] == 'M' or ner_split_next[1][0] == "E"):
                                outp.write("[" + read_list[line][word] + ' ')
                            else:
                                outp.write("[" + read_list[line][word] + "]" + NER + ' ')

                    elif S == "M":
                        print(read_list[line][word], ner_list[line][word])
                        outp.write(read_list[line][word] + ' ')

                    elif S == "E":
                        outp.write(read_list[line][word] + ']' + NER + ' ')

                elif len(ner_split) == 2:
                    S1 = ner_split[0][0]
                    S2 = ner_split[1][0]
                    NER1 = ner_split[0][2:]
                    NER2 = ner_split[1][2:]

                    if S1 == "B" and S2 == "B":  # [[key]bod ...]ner
                        ner_split_next = ner_list[line][word + 1].split('/')

                        if len(ner_split_next) == 1:

                            if ner_list[line][word + 1][0] != 'M' and ner_list[line][word + 1][0] != 'E':
                                outp.write("[[" + read_list[line][word] + "]" + NER1 + ']' + NER2 + ' ')
                            else:
                                outp.write("[[" + read_list[line][word] + "]" + NER1 + ' ')

                        elif len(ner_split_next) == 2:
                            if ner_split_next[0][0] != 'M' and ner_split_next[0][0] != 'E':
                                outp.write("[[" + read_list[line][word] + "]" + NER1 + ']' + NER2 + ' ')
                            else:
                                outp.write("[[" + read_list[line][word] + ' ')

                    elif S1 == "B" and S2 == "M":  # [...[key]bod ...]ner:
                        ner_split_next = ner_list[line][word + 1].split('/')

                        if len(ner_split_next) == 1:
                            if ner_list[line][word + 1][0] != 'M' and ner_list[line][word + 1][0] != 'E':
                                outp.write("[" + read_list[line][word] + "]" + NER1 + ']' + NER2 + ' ')
                            else:
                                outp.write("[" + read_list[line][word] + "]" + NER1 + ' ')

                        elif len(ner_split_next) == 2:
                            if ner_split_next[0][0] != 'M' and ner_split_next[0][0] != 'E':
                                outp.write("[" + read_list[line][word] + "]" + NER1 + ']' + NER2 + ' ')
                            else:
                                outp.write("[" + read_list[line][word] + ' ')

                    elif S1 == "B" and S2 == "E":  # [...[key]bod]ner:
                        ner_split_next = ner_list[line][word + 1].split('/')
                        outp.write("[" + read_list[line][word] + "]" + NER1 + ']' + NER2 + ' ')

                    elif S1 == "M" and S2 == "B":
                        # 沒有這種情況
                        pass
                    elif S1 == "M" and S2 == "M":  # [...[key key2 key3]bod]ner
                        outp.write(read_list[line][word] + ' ')
                    elif S1 == "M" and S2 == "E":
                        # 沒有這種情況
                        pass
                    elif S1 == "E" and S2 == "B":
                        # 沒有這種情況
                        pass
                    elif S1 == "E" and S2 == "M":  # [...[key key2 key3]bod...]ner
                        outp.write(read_list[line][word] + ']' + NER1 + ' ')
                    elif S1 == "E" and S2 == "E":  # [...[key key2 key3]bod]ner
                        outp.write(read_list[line][word] + ']' + NER1 + "]" + NER2 + ' ')

            outp.write('\n')
```

图七、NER 文本还原并输出原格式 txt 文档核心代码。

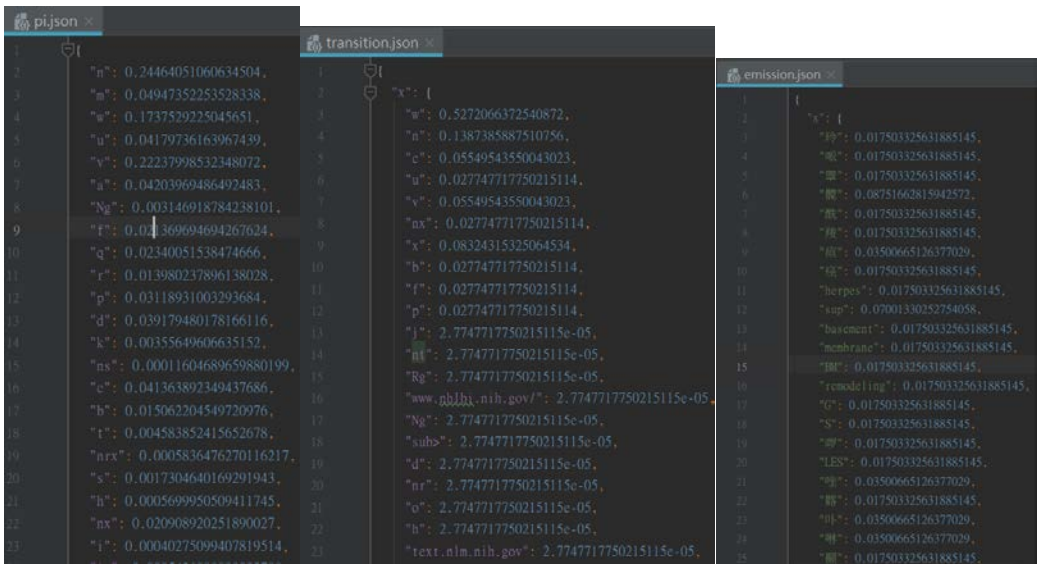
(5) evaluate.py 载入 (2) 中的给定评估用语料输出的 json 档 target.json 比对 (4) 的经此次作业算法输出的 tagged.json 后，对模型标注准确度进行评估，并输出准确率与混淆矩阵图 (**图十六、十七**)。

四、实验结果:

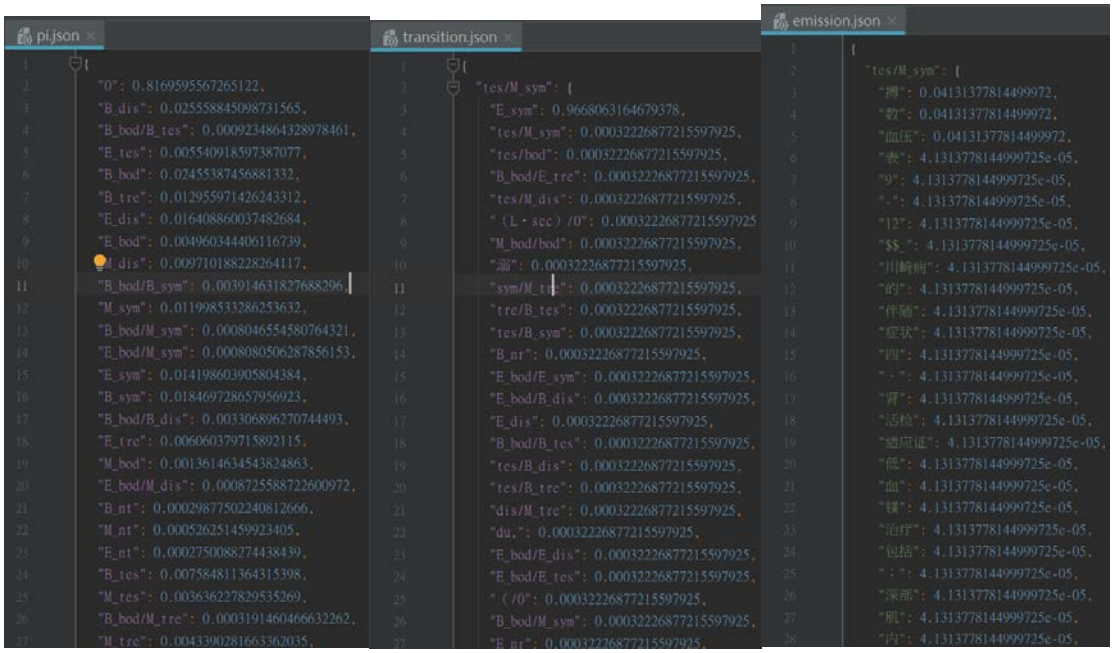
HMM 分词模型的初始概率分布π、转移概率分布 A、观测(发射)概率分布 B 如**图八~十**所示，另我们对实验结果的分词、NER、POS 任务分别进行准确率 (accuracy)、精确率 (含 Word Precison、Tag precision)，recall (含 Word Recall、Tag recall)，F-measure (含 Word F-measure、Tag F-measure) 、F1-score 、support 五项监督学习常用的评估指标的计算输出，计算公式如下**图十一**所示。



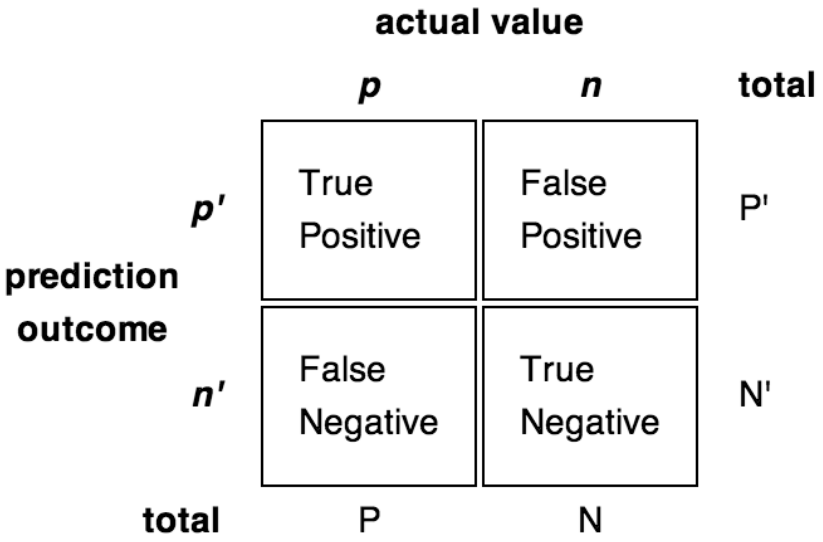
图八、分词模型的初始概率分布π、转移概率分布 A、观测(发射)概率分布 B *部分截图。*



图九、POS 模型的初始概率分布π、转移概率分布 A、观测(发射)概率分布 B *部分截图。*



图十、NER 模型的初始概率分布π、转移概率分布 A、观测(发射)概率分布 B *部分截图。*



Performance measures

accuracy (ACC)

- $ACC = (TP + TN)/(P + N)$

precision = positive predictive value (PPV)

- $PPV = TP/(TP + FP)$

recall = sensitivity = true positive rate (TPR)

- $TPR = TP/P = TP/(TP + FN)$

false positive rate (FPR)

- $FPR = FP/N = FP/(FP + TN)$

specificity (SPC) = True Negative Rate

- $SPC = TN/N = TN/(FP + TN) = 1 - FPR$

negative predictive value (NPV)

- $NPV = TN/(TN + FN)$

false discovery rate (FDR)

- $FDR = FP/(FP + TP) = 1 - PPV$

F1 score = harmonic mean of precision and recall

- $F1 = 2TP/(P + P') = 2TP/(2TP + FP + FN)$

图十一、准确度、recall、f1-score、support 四项评估指标的计算公式[13]

以 POS 准确率计算公式为例，**标记后的测试集词性相符于发放的已标计测试集(test_pos1.txt)字词数量 / 测试集字词总数量(即正确被标记与非正确被标记的字词数和) * 100%。**

下表二 及 下表三 为本次报告模型计算结果的各项输出评估指标，表二与表三的差异为**表二**为使用助教提供之语料集的分词结果去进行标注POS 与NER；**表三**为使用我组模型输出的分词结果进行标注。另因客观条件限制，有些指标只对 POS 模型有所输出，未另行解释的为使用助教所提供评估程式所输出结果，下划线部分为我组使用自行设计或使用网上开源评估模型所输出之结果，其中 NER 部分由于我组使用自研之标注方式，所以使用助教提供之评估程式进行计算的精确率较低，然而使用网上评估模型的输出结果可达 80% 左右，于 **五、误差分析**，将有详细介绍。

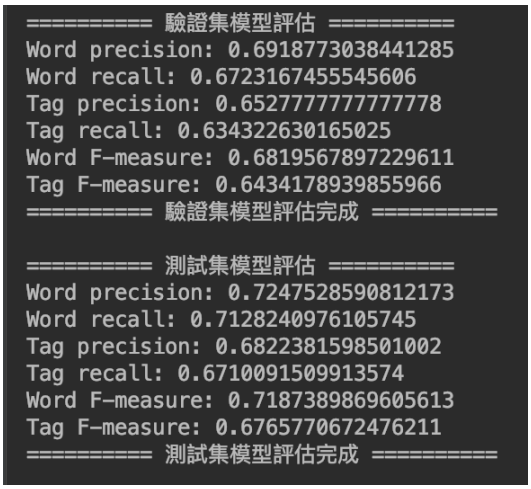
任务	Word Precison	Word Recall	Tag precision	Tag recall	Word F-measure	Tag F-measure	F1-score	support
POS 验证集	0.9356	0.9150	0.8675	0.8485	0.9252	0.8579	0.92	15192
POS 测试集	0.9894	0.9676	0.9192	0.8990	0.9784	0.9090	0.95	15476
NER 验证集	0.5187 0.8096	0.4463	N/A	N/A	N/A	N/A	0.4798 0.81	15193
NER 测试集	0.5169 0.8473	0.4631	N/A	N/A	N/A	N/A	0.4885 0.85	15476

表二、各实验结果输出评估指标(使用助教提供的分词语料集)。下划线代表使用非助教提供之评估模型计算得到，N/A 表未计算该指标或不需计算。

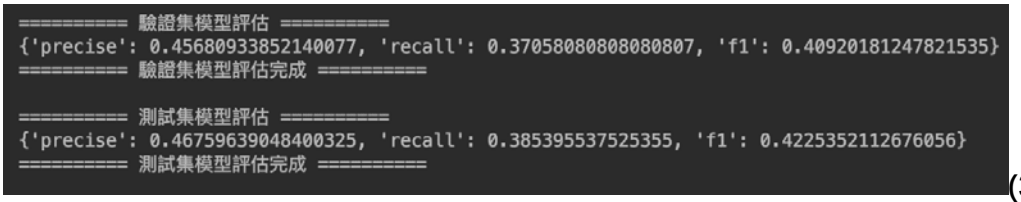
任务	Word Precison	Word Recall	Tag precision	Tag recall	Word F-measure	Tag F-measure	F1-score	support
分词 验证集	0.7955	0.7955	N/A	N/A	0.7955	N/A	N/A	N/A
分词 测试集	0.8026	0.8026	N/A	N/A	0.8026	N/A	N/A	N/A
POS 验证集	0.6918	0.6723	0.6527	0.6343	0.6819	0.6434	N/A	N/A
POS 测试集	0.7247	0.7128	0.6822	0.6710	0.7187	0.6765	N/A	N/A
NER 验证集	0.4568	0.3705	N/A	N/A	N/A	N/A	0.4092	N/A
NER 测试集	0.4676	0.3854	N/A	N/A	N/A	N/A	0.4225	N/A

表三、各实验结果输出评估指标(使用我组的分词输出结果)。下划线代表使用非助教提供之评估模型计算得到，N/A 表未计算该指标或不需计算。

(1)

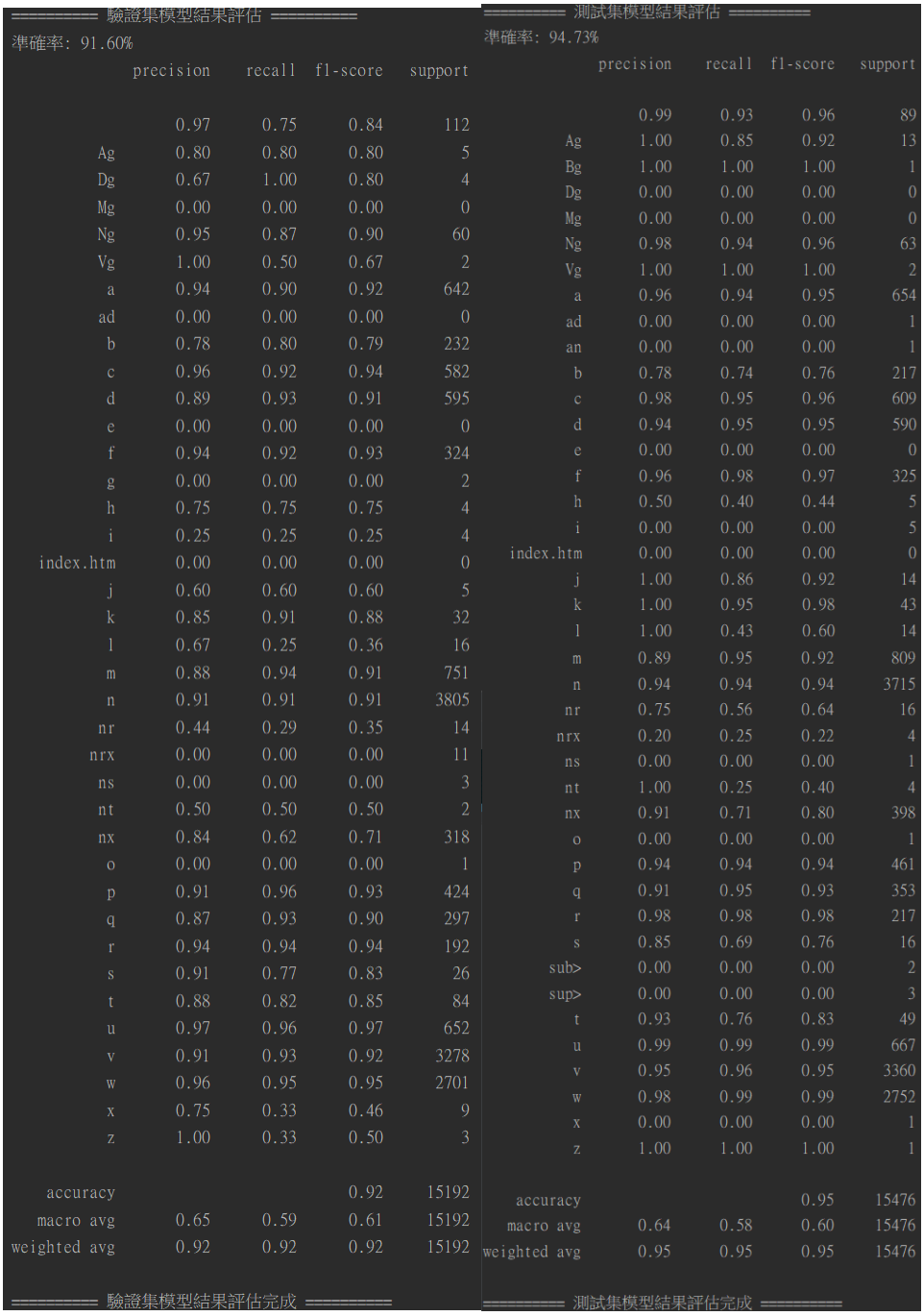


(2)



(3)

图十二、(1)分词任务 (2) POS (3) NER 验证集与测试集结果评估输出(使用自行分词训练集)。包括了 Precison、recall、F measure、花费时间 (sec)



========= 驗證集模型結果評估 =========

準確率: 91.60%

	precision	recall	f1-score	support
	0.97	0.75	0.84	112
Ag	0.80	0.80	0.80	5
Dg	0.67	1.00	0.80	4
Mg	0.00	0.00	0.00	0
Ng	0.95	0.87	0.90	60
Vg	1.00	0.50	0.67	2
a	0.94	0.90	0.92	642
ad	0.00	0.00	0.00	0
b	0.78	0.80	0.79	232
c	0.96	0.92	0.94	582
d	0.89	0.93	0.91	595
e	0.00	0.00	0.00	0
f	0.94	0.92	0.93	324
g	0.00	0.00	0.00	2
h	0.75	0.75	0.75	4
i	0.25	0.25	0.25	4
index.htm	0.00	0.00	0.00	0
j	0.60	0.60	0.60	5
k	0.85	0.91	0.88	32
l	0.67	0.25	0.36	16
m	0.88	0.94	0.91	751
n	0.91	0.91	0.91	3805
nr	0.44	0.29	0.35	14
nrx	0.00	0.00	0.00	11
ns	0.00	0.00	0.00	3
nt	0.50	0.50	0.50	2
nx	0.84	0.62	0.71	318
o	0.00	0.00	0.00	1
p	0.91	0.96	0.93	424
q	0.87	0.93	0.90	297
r	0.94	0.94	0.94	192
s	0.91	0.77	0.83	26
t	0.88	0.82	0.85	84
u	0.97	0.96	0.97	652
v	0.91	0.93	0.92	3278
w	0.96	0.95	0.95	2701
x	0.75	0.33	0.46	9
z	1.00	0.33	0.50	3
accuracy			0.92	15192
macro avg	0.65	0.59	0.61	15192
weighted avg	0.92	0.92	0.92	15192

========= 驗證集模型結果評估完成 =========

========= 測試集模型結果評估 =========

準確率: 94.73%

	precision	recall	f1-score	support
	0.99	0.93	0.96	89
Ag	1.00	0.85	0.92	13
Bg	1.00	1.00	1.00	1
Dg	0.00	0.00	0.00	0
Mg	0.00	0.00	0.00	0
Ng	0.98	0.94	0.96	63
Vg	1.00	1.00	1.00	2
a	0.96	0.94	0.95	654
ad	0.00	0.00	0.00	1
an	0.00	0.00	0.00	1
b	0.78	0.74	0.76	217
c	0.98	0.95	0.96	609
d	0.94	0.95	0.95	590
e	0.00	0.00	0.00	0
f	0.96	0.98	0.97	325
h	0.50	0.40	0.44	5
i	0.00	0.00	0.00	5
index.htm	0.00	0.00	0.00	0
j	1.00	0.86	0.92	14
k	1.00	0.95	0.98	43
l	1.00	0.43	0.60	14
m	0.89	0.95	0.92	809
n	0.94	0.94	0.94	3715
nr	0.75	0.56	0.64	16
nrx	0.20	0.25	0.22	4
ns	0.00	0.00	0.00	1
nt	1.00	0.25	0.40	4
nx	0.91	0.71	0.80	398
o	0.00	0.00	0.00	1
p	0.94	0.94	0.94	461
q	0.91	0.95	0.93	353
r	0.98	0.98	0.98	217
s	0.85	0.69	0.76	16
sub>	0.00	0.00	0.00	2
sup>	0.00	0.00	0.00	3
t	0.93	0.76	0.83	49
u	0.99	0.99	0.99	667
v	0.95	0.96	0.95	3360
w	0.98	0.99	0.99	2752
x	0.00	0.00	0.00	1
z	1.00	1.00	1.00	1
accuracy			0.95	15476
macro avg	0.64	0.58	0.60	15476
weighted avg	0.95	0.95	0.95	15476

========= 測試集模型結果評估完成 =========

图十三、POS 验证集与测试集结果评估输出(使用助教提供之已分词训练集)。包括了准确率、recall、f1-score、support。

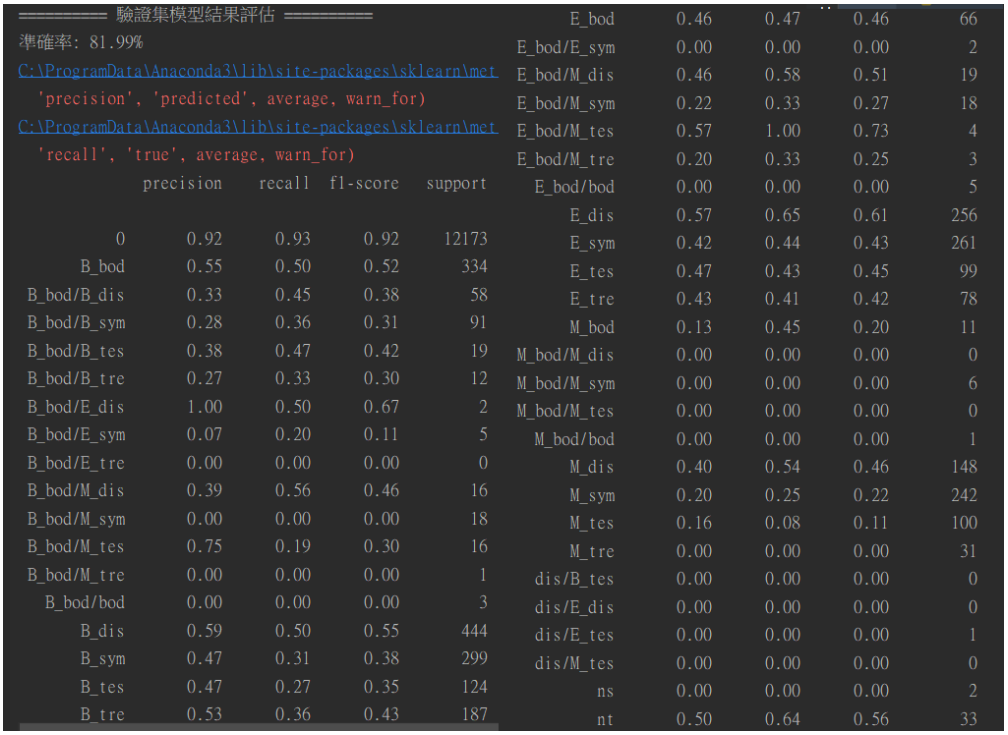
========== 測試集模型結果評估 ==========

準確率: 84.07%

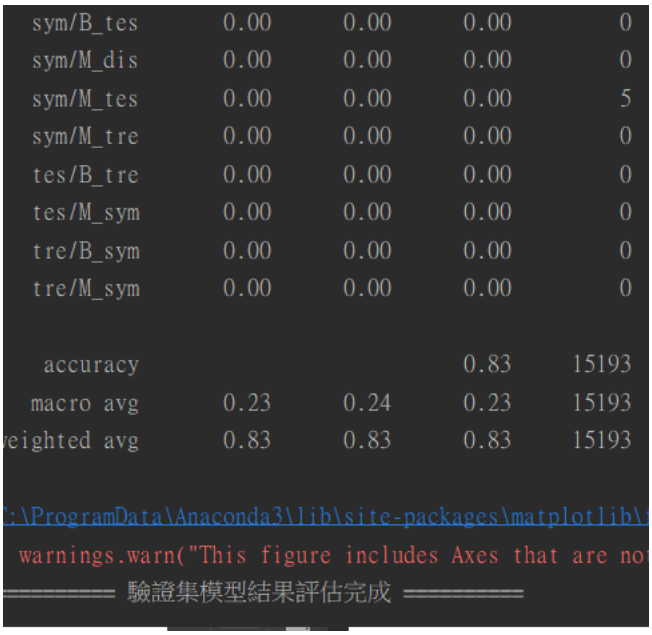
	precision	recall	f1-score	support
0	0.93	0.94	0.93	12616
B_bod	0.63	0.50	0.56	436
B_bod/B_dis	0.45	0.62	0.52	53
B_bod/B_sym	0.28	0.37	0.32	62
B_bod/B_tes	0.15	0.33	0.21	9
B_bod/B_tre	0.27	0.54	0.36	13
B_bod/E_dis	0.00	0.00	0.00	4
B_bod/E_sym	0.00	0.00	0.00	1
B_bod/E_tes	0.00	0.00	0.00	5
B_bod/E_tre	0.00	0.00	0.00	3
B_bod/M_dis	0.40	0.32	0.35	19
B_bod/M_sym	0.27	0.17	0.21	18
B_bod/M_tes	0.33	0.12	0.18	8
B_bod/M_tre	0.00	0.00	0.00	3
B_bod/bod	0.00	0.00	0.00	1
B_dis	0.63	0.54	0.59	381
B_sym	0.52	0.36	0.42	259
B_tes	0.50	0.33	0.40	145
B_tre	0.44	0.39	0.41	152
E_bod	0.45	0.38	0.41	102
E_bod/E_sym	0.00	0.00	0.00	1
E_bod/E_tes	0.00	0.00	0.00	1
E_bod/E_tes	0.00	0.00	0.00	1
E_bod/E_tre	0.00	0.00	0.00	0
E_bod/M_dis	0.64	0.67	0.65	21
E_bod/M_sym	0.38	0.60	0.46	15
E_bod/M_tes	0.00	0.00	0.00	4
E_bod/M_tre	0.75	0.75	0.75	4
E_bod/bod	0.00	0.00	0.00	1
E_dis	0.66	0.68	0.67	257
E_sym	0.44	0.44	0.44	217
E_tes	0.50	0.47	0.48	98
E_tre	0.37	0.50	0.42	74
M_bod	0.21	0.10	0.13	41
M_bod/B_dis	0.00	0.00	0.00	1
M_bod/M_dis	0.00	0.00	0.00	1
M_bod/M_sym	0.00	0.00	0.00	1
M_bod/M_tes	0.00	0.00	0.00	0
M_bod/M_tre	0.00	0.00	0.00	0
M_dis	0.51	0.43	0.46	152
M_sym	0.27	0.27	0.27	177
M_tes	0.15	0.14	0.15	70
M_tre	0.20	0.42	0.27	43
dis/B_tes	0.00	0.00	0.00	1
dis/E_dis	0.00	0.00	0.00	0
dis/M_tes	0.00	0.00	0.00	1
nt	0.19	1.00	0.32	3
sym/B_tes	0.00	0.00	0.00	0
sym/M_dis	0.00	0.00	0.00	0
tes/B_tre	0.00	0.00	0.00	0
tes/M_sym	0.00	0.00	0.00	0
tes/bod	0.00	0.00	0.00	0
tre/B_dis	0.00	0.00	0.00	1
tre/B_sym	0.00	0.00	0.00	0
tre/B_tre	0.00	0.00	0.00	1
tre/M_sym	0.00	0.00	0.00	0
accuracy			0.85	15476
macro avg	0.21	0.22	0.21	15476
weighted avg	0.85	0.85	0.85	15476

========== 測試集模型結果評估完成 ==========

图十四、NER 测试集结果评估输出(使用助教提供之已分词训练集)。包括了准确率、recall、f1-score、support。

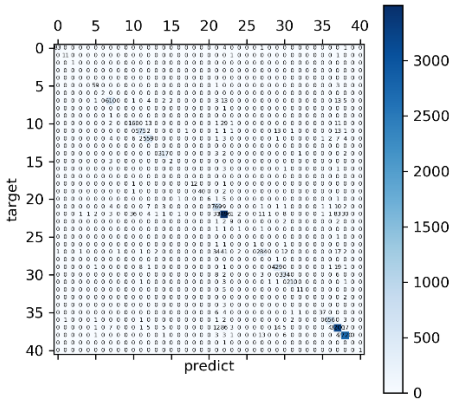


```
========== 驗證集模型結果評估 ==========
準確率: 81.99%
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\met
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\met
  'recall', 'true', average, warn_for)
              precision    recall  f1-score   support

           0       0.92      0.93      0.92     12173
       B_bod       0.55      0.50      0.52       334
   B_bod/B_dis       0.33      0.45      0.38        58
   B_bod/B_sym       0.28      0.36      0.31        91
   B_bod/B_tes       0.38      0.47      0.42        19
   B_bod/B_tre       0.27      0.33      0.30        12
   B_bod/E_dis       1.00      0.50      0.67         2
   B_bod/E_sym       0.07      0.20      0.11         5
   B_bod/E_tre       0.00      0.00      0.00         0
   B_bod/M_dis       0.39      0.56      0.46        16
   B_bod/M_sym       0.00      0.00      0.00        18
   B_bod/M_tes       0.75      0.19      0.30        16
   B_bod/M_tre       0.00      0.00      0.00         1
     B_bod/bod       0.00      0.00      0.00         3
       B_dis       0.59      0.50      0.55       444
       B_sym       0.47      0.31      0.38       299
       B_tes       0.47      0.27      0.35       124
       B_tre       0.53      0.36      0.43       187
       E_bod       0.46      0.47      0.46        66
   E_bod/E_sym       0.00      0.00      0.00         2
   E_bod/M_dis       0.46      0.58      0.51        19
   E_bod/M_sym       0.22      0.33      0.27        18
   E_bod/M_tes       0.57      1.00      0.73         4
   E_bod/M_tre       0.20      0.33      0.25         3
     E_bod/bod       0.00      0.00      0.00         5
       E_dis       0.57      0.65      0.61       256
       E_sym       0.42      0.44      0.43       261
       E_tes       0.47      0.43      0.45        99
       E_tre       0.43      0.41      0.42        78
       M_bod       0.13      0.45      0.20        11
   M_bod/M_dis       0.00      0.00      0.00         0
   M_bod/M_sym       0.00      0.00      0.00         6
   M_bod/M_tes       0.00      0.00      0.00         0
     M_bod/bod       0.00      0.00      0.00         1
       M_dis       0.40      0.54      0.46       148
       M_sym       0.20      0.25      0.22       242
       M_tes       0.16      0.08      0.11       100
       M_tre       0.00      0.00      0.00        31
   dis/B_tes       0.00      0.00      0.00         0
   dis/E_dis       0.00      0.00      0.00         0
   dis/E_tes       0.00      0.00      0.00         1
   dis/M_tes       0.00      0.00      0.00         0
          ns       0.00      0.00      0.00         2
          nt       0.50      0.64      0.56        33
```

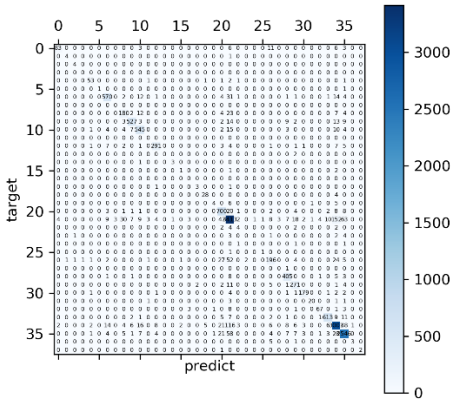


```
   sym/B_tes       0.00      0.00      0.00         0
   sym/M_dis       0.00      0.00      0.00         0
   sym/M_tes       0.00      0.00      0.00         5
   sym/M_tre       0.00      0.00      0.00         0
   tes/B_tre       0.00      0.00      0.00         0
   tes/M_sym       0.00      0.00      0.00         0
   tre/B_sym       0.00      0.00      0.00         0
   tre/M_sym       0.00      0.00      0.00         0

    accuracy                           0.83     15193
   macro avg       0.23      0.24      0.23     15193
eighted avg       0.83      0.83      0.83     15193

:\ProgramData\Anaconda3\lib\site-packages\matplotlib\
 warnings.warn("This figure includes Axes that are no
========== 驗證集模型結果評估完成 ==========
```

图十五、NER 验证集结果评估输出(使用助教提供之已分词训练集)。包括了准确率、recall、f1-score、support。

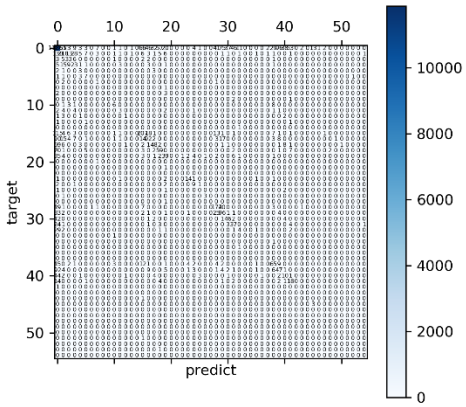


(1) 测试集

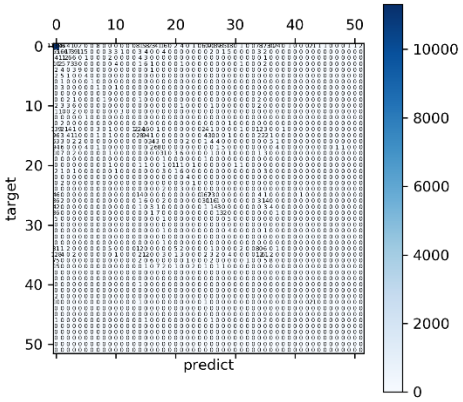


(2) 验证集

图十六、POS 实验结果混淆矩阵。



(1) 测试集



(2) 验证集

图十七、NER 实验结果混淆矩阵。

五、误差分析:

于词性切分中，我们发现在训练集中有着以下 **(表四)** 未标注或难以处理的的无意义词:

text.nlm.nih.gov	www.nhlbi.nih.gov	index.htm
guides.htm	zhyxh.htm	sup>Tc

表四、训练集中无意义词

是故我们将其手动删除，以免加入词性转移概率字典后，影响往后模型训练结果。

POS 的测试集中，我们也发现虽然提供的语料库 test_pos.txt 与 test_cwt.txt 中有不一致的情形，此情况虽然为偶见，但仍不确地其对标注准确率的影响。NER 的训练集中，也常出现词性标注的 “[” 或 “]” 出现不成对的情况，导致程序判断出错。

没有在训练集出现的未登录词，于概率计算中可能会有缺失的问题，因此我们对于此类未登录词我们对其做默认名词的设定，并且将其出现机率预设为 1e-4，并将其放入放射矩阵中做相关计算。

在语句切分中，我们尝试将训练及语料及测试集语料都合并为同一 diction 中，之后再以分号/句号/换行符做切分，作为模型的输入。但有时会因单一句子段落过长(100个字以上)，此种过长句子段直接于模型中输入，会造成转移矩阵与放射矩阵相乘后机率过低的问题，是故过长句子仍需对其做切分处理，而根据我们的测试结果，在 POS 验证集 50字 / 测试集 100字、NER 验证集 50 字 / 测试集 80 字 左右做切分会有着较为良好的效果，长度的变化因该语料集的长度而异。因为在 python 中可计算的最小数为 2.2250738585072014e-308，所以在维特比算法的标注过程中，A, B, π，三要素的相乘过程会使得最后标注的机率过低 (无法比较各项最佳词性标注路径机率大小)，在矩阵计算的最后会趋近于零而影响最终标注结果。

在 POS 标注中，我们尝试在读取数据集的时候，遇到英文字词就将其标注为 nx，遇到 $$_ 就将其消除，以提高词性标注的精确率，但因英文字词与 $$_ 因其在文本的比例不高，是故对精确率并无明显提高。

于 NER 标注中，我们尝试进行以下优化以提高精确率:

- 增加 ner 中的 nt, nr 标注判断
- 验证集 line 150 肾衰竭在 cws 中没切分成「肾 衰竭」导致行数不一致
- 为提高精确率，在 helper/pre_proccess.py 的标注方法新增 tag_level 参数，原先的ner 有三级 (B, M, E)，尝试改为二级(B, E)（使验证集/测试集的准确率皆上升 1%）。

于助教提供的评估程式中，我组的 NER 评估精确率约为 50％ 左右，如表二所示，然而另外使用我组自研的分词标注方式，可以使精确率提高至 80％ 左右，我组的标注 tag 如下表所示。规则为 B 前缀表示命名实体标注的第一个字，M 前缀表示中间词，E 前缀表示标注最末的词。。

B_bod	E_bod	M_bod
B_dis	E_dis	M_dis
B_tes	E_tes	M_tes
B_tre	E_tre	M_tre
B_sym	E_sym	M_sym
B_bod/M_sym	E_bod/M_sym	M_bod/M_tre
B_bod/B_tes	E_bod/M_dis	M_bod/M_tes
B_bod/B_sym	E_bod/M_tes	M_bod/M_sym
B_nt	E_nt	M_nt
B_bod/B_dis	E_bod/E_sym	M_bod/M_sym
B_bod/M_tre	E_bod/M_tre	M_bod/M_dis
B_bod/E_dis	E_bod/E_dis	M_bod/B_dis
B_bod/E_tre	E_bod/B_sym	
B_bod/M_tes	E_bod/E_tes	
B_bod/B_tre		
B_bod/M_dis		
B_bod/E_sym		
B_bod/E_tes		
B_bod/M_nt		

表五、NER 标注嵌套规则。使用该规则作为发散与转移概率模型的参数。

在一般的命名实体标注中，不存在双层嵌套的情况，以人民日报语料库中的一个命名实体标注为例：[亚/j 太/j 经合/j 组织/n]nt，若将其转换为（B、

M、E）格式则为：亚/B_nt 太/M_nt 经合/M_nt 组织/E_nt。然而一个词配一个标注的形式无法顺理解决一个词同属两个命名实体的情况，所以我们将同属两个 ner 的词直接打上两个标注，再将其两个标注视为一个特征进行训练。最后再通过程序输出成原本的格式。

以 [[黏膜]bod 肿胀]sym 为例，在这里，「粘膜」这个词同时属于 bod 和 sym 标注，代表「粘膜」这个词是 bod 命名实体中的首词；同时也是 sym 中的首词。我们将该词的标注通过程序转换为「粘膜/B_bod/B_sym 肿胀/E_sym」，再以第一个“/”切分该词，将其后面所有的标注作为该词的标签并用于训练的输入。且根据隐马尔可夫的算法规则，不需担心在预测步骤中属于同一命名实体标注的词，其 E 或 M 在 B 之前的情况，在计算 ner 的转移概率矩阵时，该标注情形出现的概率会极低且趋近于零。

六、思考题:

根据这次作业提出几个开放性问题，供学生思考；大家可以把这部分体现在实验报告或代码中，每道题酌情加 0-10 分

1. NER 部分存在同一个词被多个 NER 嵌套的情况，但是序列标注的 NER 模型对于一个词往往只能有一个 NER 标注，如何解决该问题？

e.g:[[肺动脉 肾小管]bod 狭窄]sym

(1) 在预处理过程中，我们将训练数据的标注方式通过程序转换为 B、M、E 加上其本身的 ner 标注（例如：B_bod、M_sym）

(2) 若一个词同属两个命名实体，则将其打上两个标注，并且规定第一个标注必须为内层 ner 标注（也就是 bod）；第二个为外层 ner 标注。

(3) 将一个词所有的命名实体标注视为一个特征，进行隐马尔可夫和维特比算法的训练和预测

范例：[[肺动脉 肾小管]bod 狭窄]sym。

在预处理过程被转换为 肺动脉/B_bod/B_sym 肾小管/E_bod/M_sym 狭窄/E_sym
用于训练的特征:

词	特征
肺动脉	B_bod/B_sym
肾小管	E_bod/M_sym
狭窄	E_sym

2. 分词任务与 pos/ner 任务其实是紧密相关的，如果在分词阶段出现错误，该部分误差就会传递下去，如何解决该问题？
专注于分词任务的完成。

3. 训练数据量十分有限，在不使用人工标注的情况下，如何扩充数据量，并进一步优化模型效果？

六、实验分工及感想:

程咏: 主要代码设计与调优、模型开发与实作、结果分析与改进、报告撰写。
詹承勋: 部分代码设计、资料搜集、误差分析、报告撰写。

由于我们都是自然语言处理界的小白，在选择算法时，为了能更好体会分词、词性标注以及命名实体标注的处理过程以及算法的思路，并且希望尽可能的不调用外部库来达成自我锻炼之目的，我们最终选择使用自然语言界的经典机器学习算法—隐马尔可夫 + 维特比来完成此次作业。

过程中花费在数据预处理的时间大概占了七成吧，尤其是在命名实体标注的训练集处理，我们觉得使用“[”和“]”对词进行标注特别不友善，也特变难处里，我们尝试写了许多处理这些词汇的程序。最后使用 stack 来处理 ner 标注，但前提必须是方括号必须是成对出现的，否则在 pop 时会造成结果异常，但我们的训练和测试数据中仍有一些 ner 标注出现格式问题，最后还是不得已通过我的玉手进行手工修改。当然在写还原成原先标注格式的程序时也花了不少时间。

在完成此次作业的过程中，也学到了许多解决问题的方法和流程。最先从隐马尔可夫算法的研究、以及加入维特比动态规划算法的代码付现；再来将数据处理成可供模型使用的格式；到最后结果输出后尝试不同办法试试看要如何使准确率进一步提升。每一个步骤中都藏着魔鬼，而我们则努力的抓出魔鬼并试着击败他。

提交了这次作业后我们也不会因此懈怠，下一步，我们会试着使用深度学习的方法再完成一次任务，并比对其中的各种差异，我们认为这才是身为一位合格工程师该有的态度，我们不盲目追求数字上的突破，而是探索数字背后所蕴含的意义。

Reference:

1. Zhang, H.-P., et al. *HHMM-based Chinese lexical analyzer ICTCLAS*. in *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*. 2003. Association for Computational Linguistics.
2. Yijia Liu, W.C., Jiang Guo, Bing Qin, Ting Liu, *Exploring Segment Representations for Neural Segmentation Models.* Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence.
3. Chang, P.-C., M. Galley, and C.D. Manning. *Optimizing Chinese word segmentation for machine translation performance*. in *Proceedings of the third workshop on statistical machine translation*. 2008. Association for Computational Linguistics.
4. Young, T., et al., *Recent trends in deep learning based natural language processing.* ieee Computational intelligenCe magazine, 2018. **13**(3): p. 55-75.
5. Ma, J., K. Ganchev, and D. Weiss, *State-of-the-art Chinese word segmentation with bi-lstms.* arXiv preprint arXiv:1808.06511, 2018.
6. Huang, Z., W. Xu, and K. Yu, *Bidirectional LSTM-CRF models for sequence tagging.* arXiv preprint arXiv:1508.01991, 2015.
7. Horsmann, T., N. Erbs, and T. Zesch. *Fast or Accurate?-A Comparative Evaluation of PoS Tagging Models*. in *GSCL*. 2015.
8. Zhou, J., W. Qu, and F. Zhang, *Chinese named entity recognition via joint identification and categorization.* Chinese journal of electronics, 2013. **22**(2): p. 225-230.
9. *结巴中文分词*. Available from: https://github.com/fxsjy/jieba.
10. *HanLP: Han Language Processing*. Available from: https://github.com/hankcs/HanLP.
11. *隐马尔可夫模型词性标注及其 Python 实现*. Available from: https://zhuanlan.zhihu.com/p/48260272.
12. 李航, *统计学习方法*. 清华大学出版社, 北京, 2012.
13. Pennsylvania, U.o. *CIS520: Machine Learning Class*. Available from: https://alliance.seas.upenn.edu/~cis520/dynamic/2018/wiki/index.php?n=Lectures.PrecisionRecall.