实 验 报 告

EXPERIMENT REPORT

目录

一 项目简介

本小组对 ACL2020 中来自复旦大学邱锡鹏老师团队的 *FLAT: Chinese NER Using Flat-Lattice Transformer* 进行论文复现，具体实现过程参考了官方开源代码 Flat-Lattice-Transformer。接下来报告将大致从两部分来介绍本项目的内容，分别是对论文的理解和项目运行过程记录。

与项目相关性较强的论文一共三篇，放在相关论文文件夹中；语料及训练集较大，上传受限，所以存放在百度云盘，后附链接；开源项目代码也已打包，放在项目代码文件夹中，后附 GitHub 链接。

语料及训练集百度云链接

链接：https://pan.baidu.com/s/1iJDo_iQnlcrMjxXXbPK-3Q

提取码：p55m

开源项目 Flat-Lattice-Transformer

链接：https://github.com/LeeSureman/Flat-Lattice-Transformer

注：若用 Jupyter Notebook 在浏览器中打开运行，直接运行 Run.ipynb 文件，V0 版本运行，进入 V0 文件，运行 Run_V0.ipynb 文件，V1 版本运行，进入 V1 文件,运行 Run_V1.ipynb 文件。用其他编译器运行可直接运行 preprocess.py 文件，以及 flat_main.py 文件，完整运行步骤见 README.MD 文件。

二 论文内容

1.基本介绍

本文在 Lattice LSTM(ACL 2018)的基础上做出了这两方面的改进：

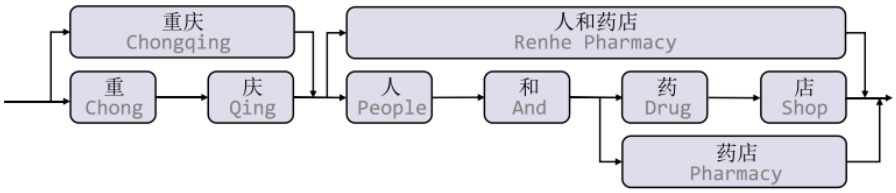
第一，作者提出了一种将 Lattice 图结构无损转换为扁平的 Flat 结构的方法，并将 LSTM 替换为了更先进的 Transformer Encoder，该方法不仅弥补了 Lattice LSTM 无法并行计算(batchsize=1)的缺陷，而且更好地建模了序列的长期依赖关系；

第二，作者提出了一种针对 Flat 结构的相对位置编码机制，使得字符与词汇得到了更充分更直接的信息交互，在基于词典的中文 NER（Named Entity Recognition，命名实体识别）模型中取得了 SOTA。

2.所作改进

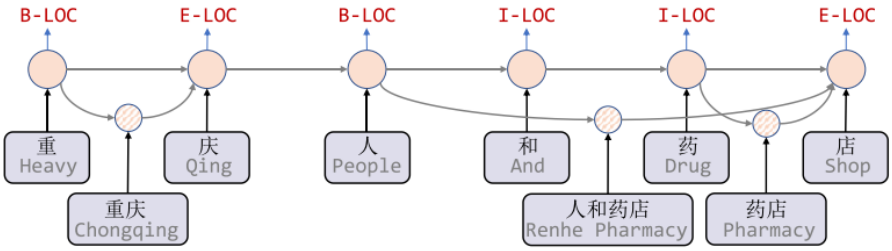
由于中文词汇的稀疏性和模糊性，基于字符的序列标注模型往往比基于词汇的序列标注模型表现更好，但在基于字符的模型中引入分词信息往往能够带来性能的提升，尤其是对于 NER 任务来说，词汇能够提供丰富的实体边界信息。Lattice LSTM 首次提出使用 Lattice 结构在 NER 任务中融入词汇信息。

汉字格结构已被证实是一种有效的中文命名实体识别方法，格子结构被证明对利用词信息和避免分词的错误传播有很大的好处。我们可以将一个句子与一个词典进行匹配，得到其中的潜词。格是一个有向无环图，其中每个节点都是一个字符或一个潜在的字。格包括句子中的一系列字符和可能的单词。它们不是按顺序排列的，单词的第一个字符和最后一个字符决定了它的位置。汉字格中的一些词可能对 NER 很重要。如图(a)所示，一个句子的 Lattice 结构是一个有向无环图，每个节点是一个字或者一个词， “人和药店(Renhe Pharmacy)”可以用来区分地理实体“重庆(Chongqing)”和“重 庆人(Chongqing People)”。



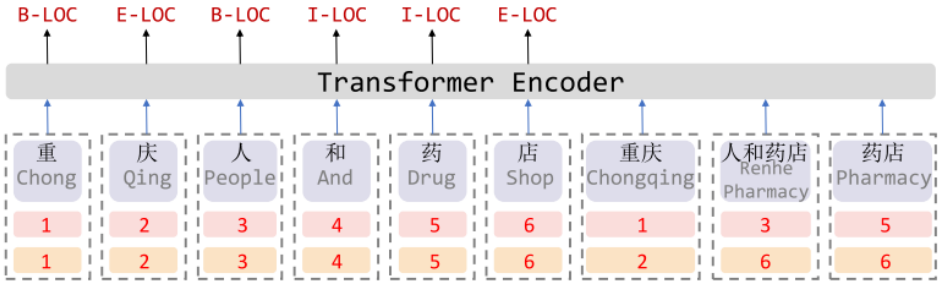
(a) Lattice.

在 Lattice LSTM 中，使用一个额外的词元对可能的词进行编码，并使用注意机制在每个位置融合可变数量的节点，如图(b)所示。LR-CNN 使用 CNN 对不同窗口大小的潜在单词进行编码。但该方法存在一定局限性，由于 Lattice 结构的动态性，Lattice LSTM 无法在 GPU 上并行训练；且 RNN 和 CNN 难以建模长距离的依赖关系，在 Lattice LSTM 中的字符只能获取前向信息，没有和词汇进行足够充分的全局交互。



(b) Lattice LSTM.

从 Transformer 的 position representation 得到启发，作者给每一个 token/span(字、词)增加了两个位置编码，分别表示该 span 在 sentence 中开始(head)和结束(tail)的位置，对于字来说，head position 和 tail position 是相同的。通过对注意力打分函数的简单改进，使得 Transformer 结构在 NER 任务上性能大幅提升，如图 (c)所示。从这样的标签序列中可以无损地重建 Lattice 结构。同时，扁平的结构允许使用 Transformer Encoder，其中的 self-attention 机制允许任何字符和词汇进行直接的交互。



(c) Flat-Lattice Transformer.

3. 数学模型

Muiti-head self-attention

有了位置编码，容易想到可以像原始 Transformer 那样将字向量直接和两个位置向量相加，然后参与后续的 self-attention：

$$\begin{aligned}\mathrm{Attn}(\mathbf{A},\mathbf{V}) &= \mathrm{softmax}(\mathbf{A})\mathbf{V};\\ \mathbf{A}_{ij} &= \left(\frac{\mathbf{Q}_i\mathbf{K}_j^{\mathrm{T}}}{\sqrt{d_{\mathrm{head}}}}\right);\\ [\mathbf{Q},\mathbf{K},\mathbf{V}] &= E_x\left[\mathbf{W}_q,\mathbf{W}_k,\mathbf{W}_v\right]\end{aligned}$$

不过这样做肯定不算是有效编码了位置信息，这也是原始 Transformer 在 NER 任务上的性能比不过 BiLSTM 的原因之一。有效的位置编码一直是改进

Transformer 的重要方向，针对本文提出的 Flat 结构，作者借鉴并优化了 Transformer-XL (ACL 2019)中的相对位置编码方法，有效地刻画了 span 之间的相对位置信息。

Relative Position Encoding of Spans

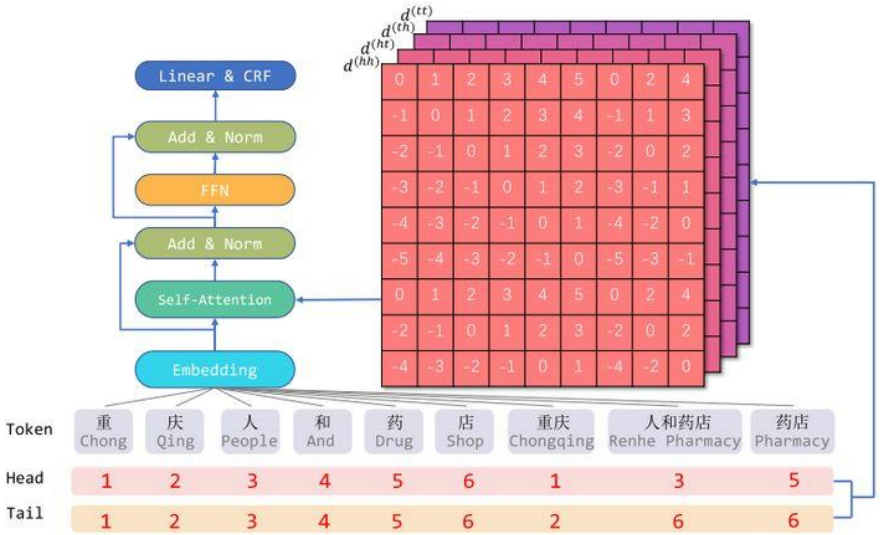


Figure 2: The overall architecture of FLAT.

span 是字符和词汇的总称，span 之间存在三种关系：交叉、包含、分离，然而作者没有直接编码这些位置关系，而是将其表示为一个稠密向量。作者用 $head[i]$和 $tail[i]$ 表示 span 的头尾位置坐标，并从四个不同的角度来计算x_i和x_j的距离：

$$
\begin{aligned}
d_{ij}^{(hh)} &= head[i] - head[j] \\
d_{ij}^{(ht)} &= head[i] - tail[j] \\
d_{ij}^{(th)} &= tail[i] - head[j] \\
d_{ij}^{(tt)} &= tail[i] - tail[j]
\end{aligned}
$$

如图 2 所示，这会得到四个相对距离矩阵：$d^{(hh)}$，$d^{(ht)}$，$d^{(th)}$，$d^{(tt)}$，其中$d_{ij}^{(hh)}$表示x_i的开始位置和x_j的开始位置的距离。然后将这四个距离拼接后作一个非线性变换，得到x_i和x_j的位置编码向量R_{ij}：

$$
R_{ij} = \mathrm{ReLU}(W_r(\mathbf{p}_{d_{ij}^{(hh)}} \oplus \mathbf{p}_{d_{ij}^{(th)}} \oplus \mathbf{p}_{d_{ij}^{(ht)}} \oplus \mathbf{p}_{d_{ij}^{(tt)}}))
$$

其中P_d是 Transfor在此处键入公式。mer 采用的绝对位置编码：

$$\mathbf{p}_d^{(2k)} = \sin\left(d/10000^{2k/d_{model}}\right)$$
$$\mathbf{p}_d^{(2k+1)} = \cos\left(d/10000^{2k/d_{model}}\right)$$

这样，每一个 span 都可以与任意 span 进行充分且直接的交互，然后作者采用了 Transformer-XL (ACL 2019)中提出的基于相对位置编码的 self-attention：

$$\mathbf{A}_{i,j}^{*} = \mathbf{W}_q^{\intercal}\mathbf{E}_{x_i}^{\intercal}\mathbf{E}_{x_j}\mathbf{W}_{k,E} + \mathbf{W}_q^{\intercal}\mathbf{E}_{x_i}^{\intercal}\mathbf{R}_{ij}\mathbf{W}_{k,R} + \mathbf{u}^{\intercal}\mathbf{E}_{x_j}\mathbf{W}_{k,E} + \mathbf{v}^{\intercal}\mathbf{R}_{ij}\mathbf{W}_{k,R}$$

可以直观地将前两项分别看作是两个 span 之间的内容交互和位置交互，后两项为全局内容和位置 bias，在 Transformer-XL 中**R**是根据绝对位置编码直接计算得出的，而这里的**R**经过了非线性变换的处理。最后，用$\mathbf{A}^{*}$替换式(1)中的**A**，取出字的编码表示，将其送入 CRF 层进行解码得到预测的标签序列。

4.结果比对

下图给出了论文的实验结果，从图中可以看出，引入词汇信息的方法，都相较于 baseline 模型 biLSTM+CRF 有较大提升。可见引入词汇信息可以有效提升中文 NER 性能。采用相同词表（词向量）时，FLAT 好于其他词汇增强方法；FLAT 如果 mask 字符与词汇间的 attention，性能下降明显，这表明 FLAT 有利于捕捉长距离依赖。且 FLAT 结合 BERT 效果会更佳。

	Lexicon	Ontonotes	MSRA	Resume	Weibo
BiLSTM	-	71.81	91.87	94.41	56.75
TENER	-	72.82	93.01	95.25	58.39
Lattice LSTM	YJ	73.88	93.18	94.46	58.79
CNNR	YJ	74.45	93.71	95.11	59.92
LGN	YJ	74.85	93.63	95.41	60.15
PLT	YJ	74.60	93.26	95.40	59.92
FLAT	YJ	**76.45**	**94.12**	**95.45**	**60.32**
FLAT_{msm}	YJ	73.39	93.11	95.03	57.98
FLAT_{mld}	YJ	75.35	93.83	95.28	59.63
CGN	LS	74.79	93.47	94.12*	63.09
FLAT	LS	**75.70**	**94.35**	**94.93**	**63.42**

	Lexicon	Ontonotes	MSRA	Resume	Weibo
BERT	-	80.14	94.95	95.53	68.20
BERT+FLAT	YJ	81.82	96.09	95.86	68.55

在推断速度方面，FLAT 论文也与其他方法进行了对比，FLAT 仅仅采用 1 层 Transformer,在指标领先的同时、推断速度也明显优于其他方法。

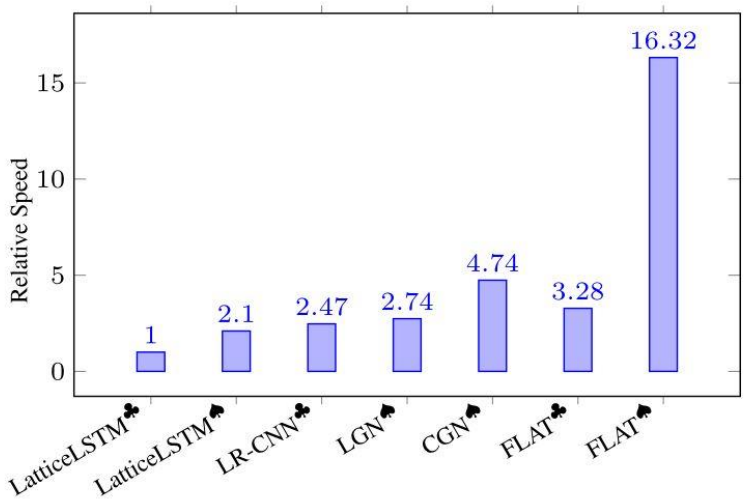


Figure 3: Inference-speed of different models, compared with lattice LSTM ♣. ♣ denotes non-batch-parallel version, and ♠ indicates the model is run in 16 batch size parallelly. For model LR-CNN, we do not get its batch-parallel version.

三 项目实践

1. 运行环境

```
Python: 3.7.3
PyTorch: 1.2.0
FastNLP: 0.5.0
Numpy: 1.16.4
```

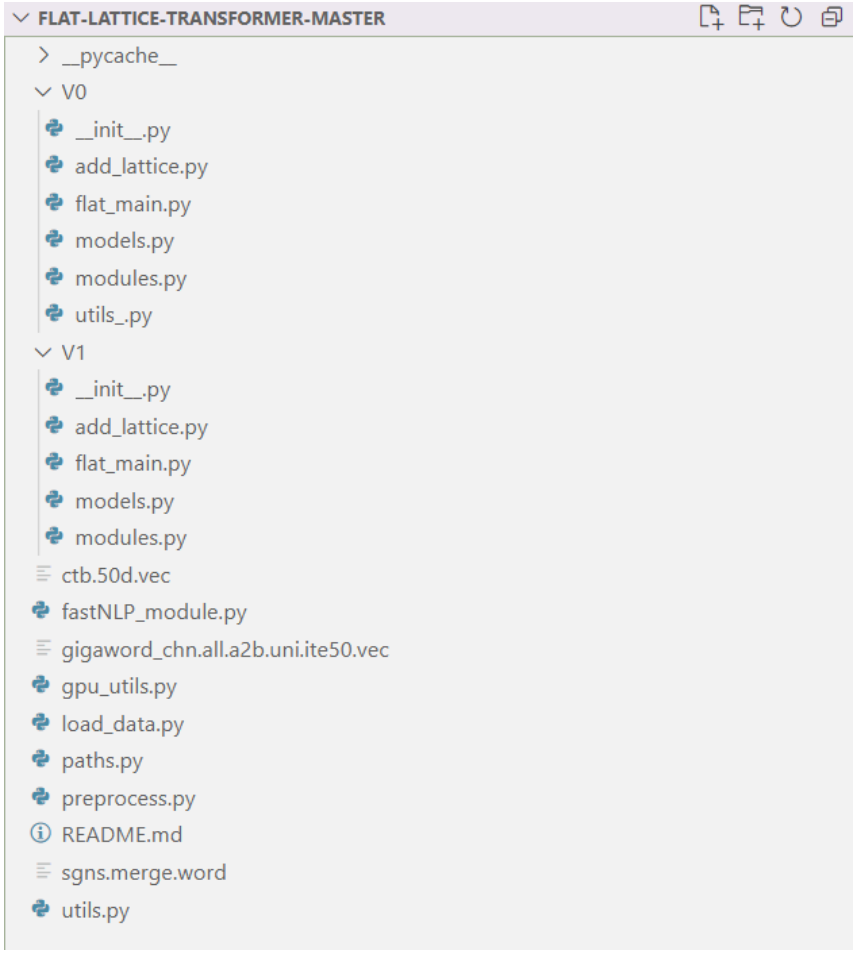
FastNLP 是作者团队自己做的一个 NLP 工具包,和本项目较贴合,使用流畅，FLAT 的代码中很多类都已定义在 FastNLP 中。与本项目匹配的是 0.5.0 的版本，建议安装这个版本，严格按照作者的指导，否则会遇到运行问题，需要重装。

2. 项目结构

1 需要的词向量文件和数据集如下

文件	默认值	解释
yangjie_rich_pretrain_unigram_path	'./gigaword_chn.all.a2b.uni.ite50.vec'	单字符的预训练vec编码
yangjie_rich_pretrain_bigram_path	'./gigaword_chn.all.a2b.bi.ite50.vec'	双字符的预训练vec编码
yangjie_rich_pretrain_word_path	'./ctb.50d.vec'	词语的预训练vec编码
yangjie_rich_pretrain_char_and_word_path	'./yangjie_word_char_mix.txt'	词语和字符混合的编码，由prepocess生成
msra_ner_cn_path	'./MSRANER'	msra数据集位置

2 项目结构



3. 参数介绍

项目中出现了大量的可调参数，且多数参数并未必给出含义的介绍，阅读代码时很影响流畅性，代码能够找到这些参数，并推断出参数的含义如下总结。

1） 数据加载参数

参数	解释	默认值
dataset	数据集名称	'msra'
bigram_min_freq	bigram编码时考虑的最小词频	1
char_min_freq	单个汉字编码时考虑的最小词频	1
word_min_freq	词语编码时考虑的最小词频	1
lattice_min_freq	添加lattice编码时考虑的最小词频	1
train_clip	是否将训练集裁剪到200以下	False
only_train_min_freq	仅对train中的词语使用min_freq筛选	True
only_lexicon_in_train	只加载在trian中出现过的词汇	False
number_normalized	0:不norm;1:char;2:char&bi;3:char&bi&lattice	0
load_dataset_seed	随机种子	100

2) bert 相关参数（仅 V1）

bert 相关的参数只有在 V1 版本中才会用到。

参数	解释	默认值
use_bert	是否使用bert编码	1
only_bert	是否只使用bert编码	0
fix_bert_epoch	多少轮之后开始训练bert	20
after_bert	如果只使用bert，bert之后的层	mlp

3) 模型参数

参数	解释	默认值
ff	feed-forward中间层的节点个数	3,修正为hidden * ff
hidden	SE位置编码和三角函数编码共用的编码维度	会修正为head_dim * head
layer	Transformer中Encoder_Layer的数量	1
head	multi-head-attn中head的个数	8
head_dim	multi-head-attn中每个head的编码维度	20
scaled	multi-head-attn中是否对attn标准化(attn_raw/sqrt(per_head_size))	False
attn_ff	是否在self-attn layer最后加一个linear层	False
ff_activate	feed-forward中的激活函数	‘relu’
use_bigram	是否使用双字符编码	1
use_abs_pos	是否使用绝对位置编码	False
use_rel_pos	是否使用相对位置编码	True
rel_pos_shared	是否共享相对位置，无效参数	True
add_pos	是否在transformer_layer中通过concat加入位置信息，无效参数	False
learn_pos	绝对和相对位置编码中编码是否可学习(是否计算梯度)	False
pos_norm	是否对位置编码进行norm(pe/pe_sum)	False
rel_pos_init	相对位置编码初始化编码方向，0: 左向右；1：右向左。无效参数	1，但是实际调用的类中写死为0
four_pos_shared	4个位置编码是不是共享权重	True
four_pos_fusion	4个位置编码融合方法’ff’, ‘attn’, ‘gate’, ‘ff_two’, ‘ff_linear’	ff_two
four_pos_fusion_shared	要不要共享4个位置融合之后形成的pos	True
k_proj	attn中是否将key经过linear层	False
q_proj	attn中是否将query经过linear层	True
v_proj	attn中是否将value经过linear层	True
r_proj	attn中是否将相对位置编码经过linear层	True
embed_dropout	embedding中的dropout	0.5
ff_dropout	ff层中的dropout	0.15
ff_dropout_2	第二个ff层中的dropout	0.15
attn_dropout	attention中的dropout	0

4）训练参数

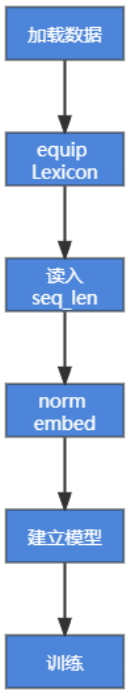
训练参数的默认值列出来后，如下。

```
epoch = 10
batch = 1
optim = 'sgd'   # sgd|adam
lr = 1e-3
warmup = 0.1
embed_lr_rate = 1
momentum = 0.9
update_every = 1
init = 'uniform'   # 'norm|uniform'
self_supervised = False
weight_decay = 0
norm_embed = True
norm_lattice_embed = True
test_batch = batch // 2
```

4. 模型

1） 模型结构

模型分为 V0 和 V1 两个版本。主要区别在于是否使用 Bert，所调用的类的名称也有所不同，但 main 脚本的流程没有太大的区别，都是按照如下的过程进行的：



首先加载数据，创建 fastNLP 中 Dataset 类型的数据集，可以通过 Dataset[‘train’]的方式去索引训练集、验证集和测试集。然后每一个数据集中，又包含如下字段。

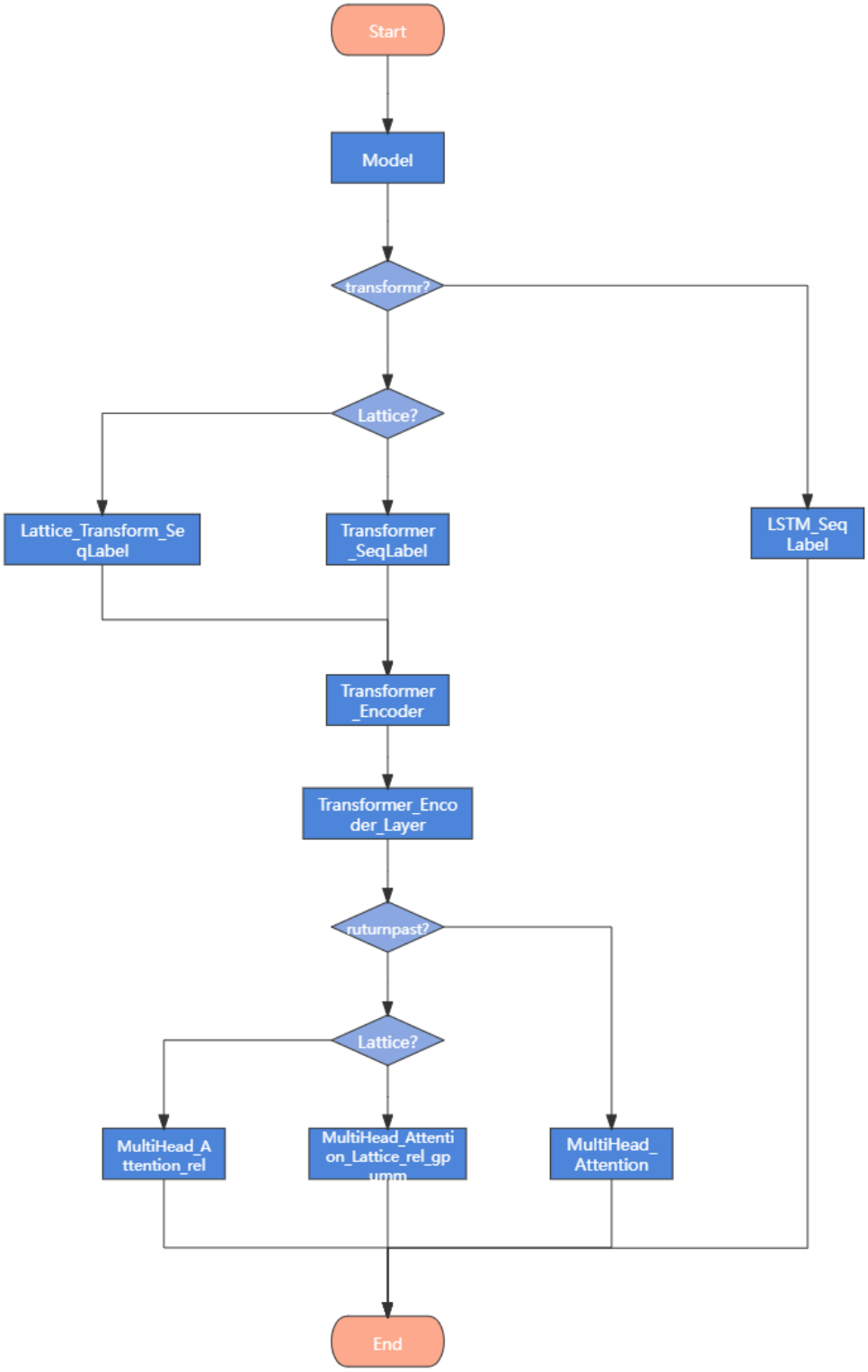
字段名	含义
char	原文本，以字为分割
target	标签
bigram	两两连续分割的字符

equip lexicon 是给数据集添加 lattice 的过程，即在词库内匹配词汇，然后放在原来的 token embedding 之后。

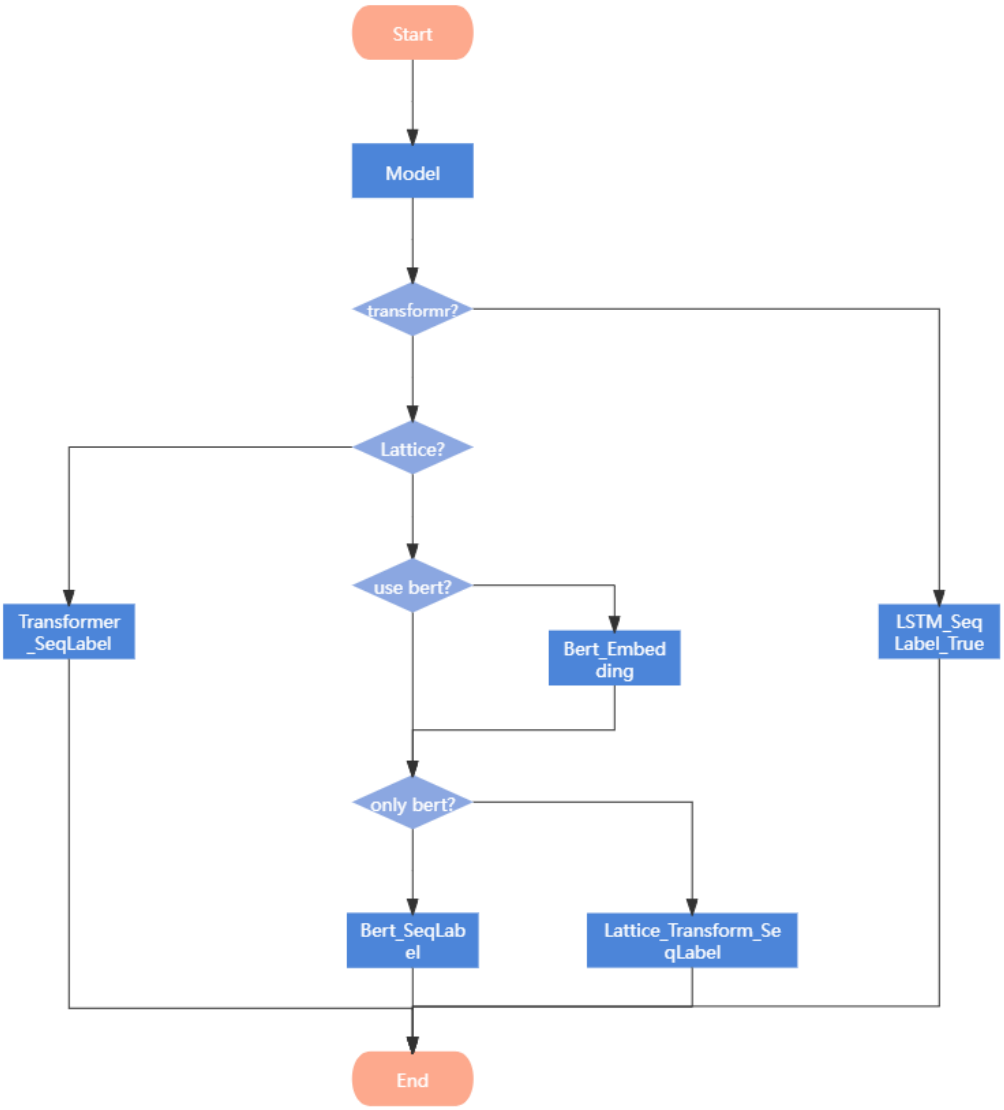
接下来将读入数据的长度，向 dataset 添加 seq_len 的字段，再对所有的 embedding 进行 normalization。

上述准备工作完成后，就会建立模型，建立哪一种模型会根据用户输入的 args 进行判断。针对不同模型，图示见后。

V0 版本的模型结构如下：



V1 版本的模型结构如下：



2） **模型训练**

以 MSRA 数据集为例，训练只需要在项目目录下执行

```
python flat_main.py --dataset msra
```

就可以根据训练参数去传参。

对遇到的问题进行总结：

1 import 报错

无法 import _get_file_name_base_on_postfix 函数。

这个错误很明显，函数名以下短线开头，不能被 import。只需要把这个函数复制到报错的脚本 load_data.py 中就可以解决。

2 ncoding_type 不对

在代码里改一下就好了，例如 MSRA 数据集，改成 bioes，CLUE 数据集，改成 bio。

3 缺少数据类型‘chain’

这个来自于\JetBrains\PyCharm\python_stubs-1902731831\itertools.py 这个脚本，把这个脚本复制到项目根路径，然后 import 它。在报错的脚本中添加：

```
from itertools import chain
```

4 Bert 相关的问题

如果是用的 V1 版本代码，可以设置使用 Bert 编码，如果使用 0.5.5 的 FastNLP 则会遇到若干问题，问题和解决方法如下。

BertModel 的名称不对：

这是因为 FastNLP 中这个类的名字变了，原本是_WordBertModel，改成了 _BertWordModel，直接复制这个类，更改名字，放入 fastNLP_Module_v1.py 里边就可以了。然后会报缺少 bert tokenizer，也是直接把 fastNLP 里边的 \modules\tokenizer\bert_tokenizer.py 里边相关的代码复制到 fastNLP_Module_v1.py 就可以解决。

5 tqdm 引发的报错

如果是在 jupyter 中执行训练，可能会由于 tqdm 版本的问题引发报错，遇到这种情况只需要把训练参数中的 use tqdm 给关掉就可以了。

3）模型保存

模型在训练的时候，利用了 fastNLP 中的一个名为 Trainer 的类，通过查看这个类的代码可以发现，这个类是已写保存方法。

只需在 flat_main.py 中，将生成 Trainer 的位置加一个参数 save_path。

```
    trainer = Trainer(datasets['train'], model, optimizer, loss,
args.batch,
                      n_epochs=args.epoch,
                      dev_data=datasets['dev'],
                      metrics=metrics,
                      device=device, callbacks=callbacks,
dev_batch_size=args.test_batch,
                      test_use_tqdm=False, check_code_level=-1,
                      update_every=args.update_every,
                      save_path='your_path/flat_lattice/{}/'.format(args
.dataset))
```

修改了之后，在保存时仍然报错，所以又对 Trainer 的 save 和 load 方法进行修改。修改后的代码如下：

```
    def _save_model(self, model, model_name):
        if self.save_path is not None:
            model_path = os.path.join(self.save_path, model_name)
            if not os.path.exists(self.save_path):
                os.makedirs(self.save_path, exist_ok=True)
            if _model_contains_inner_module(model):
                model = model.module
            torch.save(model.state_dict(), model_path)  # 只改了这一行

    def _save_model(self, model, model_name):
        if self.save_path is not None:
            model_path = os.path.join(self.save_path, model_name)
            model.load_state_dict(torch.load(model_path))
        elif hasattr(self, "_best_model_states"):
            model.load_state_dict(self._best_model_states)
        else:
            return False
        return True
```

这样改好了之后，执行 flat_main.py 脚本进行训练之后，就会在 save_path 路径下保存一个模型权重文件。

4） **模型加载**

模型加载很简单，只需要在 flat_main.py 中，实例化 model 之后，load 之前保存的权重文件。

```
    model_path = '/msra/best_Lattice_Transformer_SeqLabel_f_2021-03-03-
14-55-31-899501'
    states = torch.load(model_path).state_dict()
    model.load_state_dict(states)
```

5） **模型预测**

作者并没有给出如何预测，但是在 fastNLP 中实际上是定义了用于预测的类的，名为 predictor，去看一下代码的话，这个类其实写的很简单，但是很实用。使用方法如下：

```
    from fastNLP.core.predictor import Predictor
    predictor = Predictor(model)   # 这里的model 是加载权重之后的model

    test_label_list =
predictor.predict(datasets['test'][:1])['pred'][0]  # 预测结果
    test_raw_char = datasets['test'][:1]['raw_chars'][0]     # 原始文字
```

上面代码中的 test_label_list 就在 test 上预测出来的 label，label 对应的 BIO 可以通过以下代码查看：

```
    for d in vocabs['label']:
        print(d)
```

这里写了一个简单的方法把 label 转换成实体（仅适用于 MSRA 数据集），如下所示：

```
def recognize(label_list, raw_chars):
    """
    根据模型预测的 label_list，找出其中的实体
    label_lsit: array
    raw_chars: list of raw_char
    return: entity_list: list of tuple(ent_text, ent_type)
    -------------
    ver: 20210303
    by: changhongyu
    """
    if len(label_list.shape) == 2:
        label_list = label_list[0]
    elif len(label_list) > 2:
        raise ValueError('please check the shape of input')
```

```
    assert len(label_list.shape) == 1
    assert len(label_list) == len(raw_chars)

    # 其实没有必要写这个
    # 但是为了将来可能适应bio的标注模式还是把它放在这里了
    starting_per = False
    starting_loc = False
    starting_org = False
    ent_type = None
    ent_text = ''
    entity_list = []

    for i, label in enumerate(label_list):
        if label in [0, 1, 2]:
            ent_text = ''
            ent_type = None
            continue
        # begin
        elif label == 10:
            ent_type = 'PER'
            starting_per = True
            ent_text += raw_chars[i]
        elif label == 4:
            ent_type = 'LOC'
            starting_loc = True
            ent_text += raw_chars[i]
        elif label == 6:
            ent_type = 'ORG'
            starting_org = True
            ent_text += raw_chars[i]
        # inside
        elif label == 9:
            if starting_per:
                ent_text += raw_chars[i]
        elif label == 8:
            if starting_loc:
                ent_text += raw_chars[i]
        elif label == 3:
            if starting_org:
                ent_text += raw_chars[i]
        # end
        elif label == 11:
            if starting_per:
```

```
                    ent_text += raw_chars[i]
                    starting_per = False
            elif label == 5:
                if starting_loc:
                    ent_text += raw_chars[i]
                    starting_loc = False
            elif label == 7:
                if starting_org:
                    ent_text += raw_chars[i]
                    starting_org = False
            elif label == 13:
                ent_type = 'PER'
                ent_text = raw_chars[i]
            elif label == 12:
                ent_type = 'LOC'
                ent_text = raw_chars[i]
            elif label == 14:
                ent_type = 'PER'
                ent_text = raw_chars[i]
            else:
                ent_text = ''
                ent_type = None
                continue

            if not (starting_per or starting_loc or starting_org) and
len(ent_text):
                # 判断实体已经结束，并且提取到的实体有内容
                entity_list.append((ent_text, ent_type))

        return entity_list

    recognize(test_label_list, test_raw_char)
    # Out:
    # [('中共中央', 'ORG'),
    #  ('中国致公党', 'ORG'),
    #  ('中国致公党', 'ORG'),
    #  ('中国共产党中央委员会', 'ORG'),
    #  ('致公党', 'ORG')]
```

5. 运行结果

V0 版本的运行结果

```
status:train
msg:_
train_clip:True
device:0
debug:0
gpumm:False
see_convergence:False
see_param:False
test_batch:5
seed:11242019
test_train:False
number_normalized:0
lexicon_name:yj
update_every:1
use_pytorch_dropout:0
char_min_freq:1
bigram_min_freq:1
lattice_min_freq:1
only_train_min_freq:True
only_lexicon_in_train:False
word_min_freq:1
epoch:100
batch:10
optim:sgd
lr:0.001
embed_lr_rate:1


self_supervised:False
weight_decay:0
norm_embed:True
norm_lattice_embed:True
warmup:0.1
use_bert:None
model:transformer
lattice:1
use_bigram:1
hidden:-1
ff:3
layer:1
head:8
head_dim:20
scaled:False
ff_activate:relu
k_proj:False
q_proj:True
v_proj:True
r_proj:True
attn_ff:False
use_abs_pos:False
use_rel_pos:True
rel_pos_shared:True
add_pos:False
learn_pos:False
pos_norm:False
rel_pos_init:1
four_pos_shared:True
four_pos_fusion:ff_two
four_pos_fusion_shared:True
pre:
post:an
```

```
post:an
embed_dropout_before_pos:False
embed_dropout:0.5
gaz_dropout:0.5
output_dropout:0.3
pre_dropout:0.5
post_dropout:0.3
ff_dropout:0.15
ff_dropout_2:0.15
attn_dropout:0
embed_dropout_pos:0
abs_pos_fusion_func:nonlinear_add
dataset:weibo
train:1350
dev:270
test:270
label_vocab:19
{0: '<pad>', 1: '<unk>', 2: 'O', 3: 'I-PER.NOM', 4: 'I-PER.NAM', 5: 'B-PER.NOM', 6: 'B-PER.NAM', 7: 'I-ORG.NAM', 8: 'I-GPE.NAM', 9: 'B-G
PE.NAM', 10: 'B-ORG.NAM', 11: 'I-LOC.NAM', 12: 'I-LOC.NOM', 13: 'I-ORG.NOM', 14: 'B-LOC.NAM', 15: 'B-LOC.NOM', 16: 'B-ORG.NOM', 17: 'B-G
PE.NOM', 18: 'I-GPE.NOM'}
Found 3294 out of 3391 words in the pre-training embedding.
```

preprocess 的导出文件

```
</s> 0.008005 0.008839 -0.007661 -0.006556 0.002733 0.006042 0.001882 0.000423 -0.007207 0.004437 -0.008713 0.002499 -0.001503 -0.001914 -0.
-0.003764 0.005159 0.006051 0.005938 0.003195 0.003090 -0.007605 -0.008192 0.009939 0.007603 0.006180 -0.001208 0.008031 -0.000990 0.001469
-0.000298 -0.005966 0.002625 -0.002675 -0.007651 0.009508 0.008759 -0.002190 -0.000452 0.001018 -0.007275 -0.008014 0.009109 0.000126 -0.005
-0.006084 -0.006153 0.003394 0.000403 0.002662
-unknown- 0.069940 -0.098932 -0.172170 0.283078 -0.304548 0.313139 -0.000668 -0.140205 -0.181840 0.084596 0.372791 0.038989 -0.243783 0.1973
-0.295537 -0.250903 0.237119 0.125302 -0.605193 0.151445 0.217743 0.244482 -0.311849 -0.228482 0.107655 -0.123164 -0.056465 -0.107158 -0.099
-0.117119 -0.093809 0.130082 -0.210286 0.105046 -0.493604 -0.500740 0.187914 0.002487 0.044111 0.144122 -0.381626 0.152143 -0.129330 -0.2686
0.182741 0.517381 0.013076 -0.070040 -0.221347 -0.305415
中国 0.451697 -0.242886 -0.262636 0.188944 0.018249 -0.076869 -0.256131 0.231452 0.146388 0.257088 -0.103044 -0.356211 -0.326923 0.055466
-0.643856 -0.537182 -0.091502 -0.259114 -0.005499 0.087779 -0.063292 -0.244018 -0.130123 -0.145545 0.430698 0.430511 -0.302934 0.288673 -0.0
-0.055082 0.162995 0.187594 -0.577868 -0.302969 -0.451103 -0.209655 0.023008 0.113147 0.263163 0.239828 -0.052018 -0.224958 -0.304611 -0.128
0.351631 0.535850 -0.230617 0.088617 -0.171322 -0.601464
记者 0.344224 -0.076145 -0.988508 1.090956 -0.503372 0.024065 0.354971 -0.624777 -0.524144 -0.302605 -1.200655 0.450614 0.024969 0.575663 0.
-1.546275 -0.548433 -0.139994 -0.787002 0.313937 -0.209102 -0.070760 0.193390 0.161052 0.137617 0.252558 -0.000460 -0.227301 -0.092986 -0.04
0.528441 -0.005232 0.111273 0.152613 -0.210307 -0.562216 0.072227 -0.144945 0.136565 0.382415 -0.548379 0.384397 -1.048480 -0.189141 -0.2056
1.916375 0.002828 0.266740 -0.100723 -0.474586
今天 0.355843 0.102234 -0.469440 0.160263 -0.135852 0.340963 0.214637 0.043403 -0.002275 0.253166 0.128384 -0.034116 -0.392530 0.092963 -0.6
-0.664588 -0.056218 0.427314 -0.532340 -0.271433 -0.204998 -0.034466 -0.501487 0.264395 0.017696 0.114779 -0.050344 0.037591 -0.038130 -0.24
-0.004371 0.103337 -0.429764 0.064783 -0.293091 -0.404938 -0.188955 -0.169577 -0.073349 -0.028268 -0.325278 0.164407 -0.441971 -0.475120 0.3
0.502502 0.081917 -0.227165 -0.661753 -0.843741
表示 0.303901 0.088188 -0.312586 0.320086 -0.190930 0.239220 0.142369 -0.115881 0.038086 0.206955 -0.141014 -0.215664 -0.357902 0.112769 -0.
-0.611672 -0.286538 0.093742 -0.179295 0.228839 -0.050416 0.221509 -0.234367 0.242233 0.112995 0.063182 0.122638 0.315353 0.158372 -0.139342
0.390293 -0.155225 -0.184150 0.074800 -0.264230 -0.318680 0.019111 -0.278691 0.106805 0.114324 -0.419585 0.217680 -0.312160 -0.520355 0.3915
0.294893 -0.078691 -0.042103 -0.458411 -0.908743
中央社 0.805398 -0.109999 -1.366091 1.795741 -1.117762 -0.223150 -0.085603 -1.751693 -0.782929 -0.014333 -1.396924 0.798207 -0.059635 1.2038
1.215974 -2.313757 -0.688513 0.609387 -1.507058 0.486653 0.141457 0.051883 0.152914 -0.543623 0.007669 -0.041373 0.509850 -0.499179 -1.02872
-0.160592 0.486190 -0.126033 0.047723 0.025965 -0.805199 -0.184409 0.211161 0.058877 0.162907 0.389729 -0.908968 0.662776 -1.164289 -0.29172
-0.472819 2.885883 0.266434 0.712784 -0.762549 -0.141300
美国 0.571098 -0.075475 0.005444 0.524472 0.205768 0.388674 -0.262589 -0.390574 -0.030203 -0.158277 -0.026587 -0.228888 -0.136400 0.131246
-0.730236 -0.262387 0.142431 -0.328008 -0.256086 0.167476 -0.406736 0.089572 -0.324592 -0.177494 0.252426 0.160235 -0.186712 0.112027 -0.043
0.268792 0.544670 0.151987 -1.114633 0.067813 -0.244463 -0.099775 -0.029501 -0.199538 -0.207253 -0.115262 -0.249708 0.166492 -0.561123 -0.55
1.023716 0.450253 -0.009849 0.230557 -0.217785 -0.200997
政府 0.610552 0.063488 -0.582808 0.373008 -0.342396 -0.162646 0.065471 -0.241984 0.059793 0.125729 -0.048453 0.044753 -0.237387 -0.300867
-0.456573 -0.427725 0.097450 -0.121185 -0.266561 0.139089 -0.013589 -0.101017 -0.276110 0.160472 0.068943 -0.102355 -0.490885 0.211651 -0.06
```

四 总结反思

成员分工

赵睿：实验报告撰写 论文搜集分析

马昌盛：代码调试和运行 论文分析

严志涵：实验报告撰写 资料收集整理

实验感想

经过这次项目我们获益匪浅，收获良多。

关于团队的优点，首先就是团队意识，在开始项目前对彼此的能力进行一定的了解，根据自己搜集的资料交流想法，也能更好地明确分工，效率也会随之提高。在这次项目中大家的发挥都非常不错，成员之间也能互相学习，互相给人的感觉也很可靠。

关于团队的不足，我们这次时间安排的不是很合理，导致我们任务完成有些粗糙，这也可能是我们第一次合作，能力有限导致的。最开始的定题也花去了不少时间。

就做项目的经验而言，我们经过讨论，一个项目首先是要有好的想法，然后才是方案设计，不要一股脑地直接就上手码代码。在实施中，不要为了学习某种方法，而去应用它； 也不要因为不熟悉，就直接舍弃某种方案，网上参考别人用过的，学习学习都能大概掌握，还能加经验值，做项目也是一个学习的过程。实施过程中遇到问题，也不能直接规避，贪图一时方便后面可能越来越复杂，最后直接掉坑里，项目的实施要以全局考虑。好的编译习惯很重要，编写代码的过程中，一串代码的往往需要反复的测试，执行，才能完善，在编译报错的情况下，又需要反复的去检查。而也有可能在编译通过之后，执行后的结果并不是想要的结果。这就可能是代码中出了其他的错误，又需要反复的检查与改动。所以，这要求我们在编写代码的过程中要更加的细心。

这次项目了解了一些课上没有学过的知识，再确定选题的过程中也对 NLP 的应用方面有了一定的了解，既增长了见识，又锻炼了能力，对小组合作的模式也更加适应。