基于关系模式约束的主体-关系-客体信息抽取

学号：16307130194，姓名：陈中钰

Abstract

本项目在具有关系约束的中文信息抽取数据集 SKE 上，构建了一个可以从短句子中提取出给定范围内的关系模式以及其主体和客体的系统。系统主要由两部分组成，分别是分类模块和序列标注模块。在输入句子后，分类模块会对句子进行多分类，给出句子中含有的关系模式，接着序列标注模块根据该关系模式，对句子进行序列标注，最后系统再从句子序列标注中提取出主体-关系-客体的三元组信息。项目实现了多个分类模型，在开发集上的 F1 值最高达到 90.30%，同时实现了两个序列标注模型，其 F1 值最高达到 85.14%，因此该系统能较好地从句子中提取出对应的关系模式及其三元组信息。

1 Introduction

信息抽取是指从自然语言文本中抽取句子关键信息，包括实体、实体关系、实体属性等实质性信息。获取文本中的关键信息不仅能帮助我们更好地理解句子，而且是智能问答、信息检索等人工智能应用的重要基础。同时，信息抽取任务可以使用量化的指标对系统效果给出评价，相对于文本生成型任务来说更容易评价系统能力的高低。因此，本项目选择了信息抽取这个任务。

为了实现从句子中提取出句子中的关系模式及其对应主体和客体，我把提取过程分为两步：第一步是从句子中提取出含有的关系模式，使用关系模式多分类 (multi-classification) 技术实现；第二步是根据关系模式，提取出关系所对应的主体和客体，通过命名实体识别 (NER)/序列标注技术来实现。项目在具有关系约束的中文信息抽取数据集 SKE 上，使用Paszke et al. [2019] 提出机器学习框架 Pytorch，训练了多个多分类模型，包括 CNN、RNN、BiLSTM、BiLSTM-pooling、RCNN 的简单分类模型，此外实现了大型预训练模型 BERT；而且，还分别实现了使用 BiLSTM、Transformer 作为编码器的序列标注模型。其中，最好的多分类模型是 BERT，在开发集上的 F1 值达到 90.30%，而最好的序列标注模型是使用 BiLSTM 作为编码器的，其 F1 值达到了 85.14%。因此，总体上来说，系统可以较好地从句子中提取出含有的关系以及对应的主体和客体。

在下文中，首先将对数据集进行基本的分析，给出任务可以分步处理的依据，并分别说明多分类和序列标注任务对应的数据集处理。接着给出系统构建中所需要的方法和技术，包括使用到的多分类、序列标注模型以及相关的算法。然后给出系统框架、模型训练的设计及相关参数、实验的设置。然后给出实验的结果，并对结果进行简单的分析。文章最后将对全文进行简短的总结，并指出项目的不足之处和可能的解决方案。

2 Dataset

具有关系约束的中文信息抽取数据集 (Schema based Knowledge Extraction, SKE)[1]给定了一系列的关系模式约束，每个关系模式`schema={S_type, P, O_type}` 定义了关系`P` 以及其对应的主体`S` 和客体`O` 的类别。给定的数据集中包括了句子文本`text`、句子对应的分词和词性列表`postag=[(word, pos), ...]`，以及句子对应的主体-关系-客体信息列

[1] http://lic2019.ccf.org.cn/kg

表spo_list=[(S_type, S, P, O_type, O), ...]。具体数据形式可以查看下方例子。其中，给出的句子的分词是不彻底的，有可能每个词语还能够继续分割为多个词语。另外，数据集还划分为训练集、开发集和测试集，其中测试集的数据是不包含 spo_list 的。

```
schemas = [
    {
        "subject_type": "影视作品",
        "predicate": "改编自",
        "object_type": "作品"
    },
    ...
]
text = "《端脑》改编自有妖气同名漫画《端脑》"
postag = [
    {
        "word": "《",
        "pos": "w"
    },
    ...
]
spo_list = [
    {
        "predicate": "改编自",
        "object_type": "作品",
        "subject_type": "影视作品",
        "object": "端脑",
        "subject": "端脑"
    },
    ...
]
```

2.1 数据分析

数据量分析 通过对 SKE 数据集进行统计可以得到，给定的关系模式有 50 种，训练集、开发集、测试集的数量如 Table 1所示，而训练集中的 text 文本长度最长为 300，所以 SKE 数据的长度是总体偏短的。

Table 1: 数据参数

train size	dev size	test size	predicate amount	max len
173108	21639	9949	50	300

spo_list 长度分析 统计数据点的 spo_list 长度，并给出训练集、开发集中各个 spo_list 长度对应的数据量占全部数据的比例，如 Figure 1所示。可以发现，无论是在训练集还是开发集中，只有约 40% 的数据是只有 1 个主体-关系-客体的三元组信息，剩下超过 50% 的文本都是拥有多个三元组信息的。通过查看数据集发现，数据中会出现同一种关系模式有多个三元组，如“某演艺作品由某人、某人和某人主演”，对应了多个“主演”关系模式，而且同一个关系模式下的主体和客体也会有不同的复杂对应关系，比如多个主体对一个客体、多个主体对多个客体等；另外，数据文本中也会同时出现多种不同的关系。

关系模式数量分析 在训练数据中对 50 种关系模式的数量进行统计，并展示出前 15 种数量最多的关系模式，如 Figure 2所示。其中 x 轴是关系模式按数据 json 文件中的顺序编号，y 轴是关系模式对应的数量。可以发现，关系模式出现的数量很不平衡，其中 47、42、31 编号的关系模式出现次数最多，分别是人物-主演-影视作品、人物-作者-图书作品、人物-歌手-歌曲。

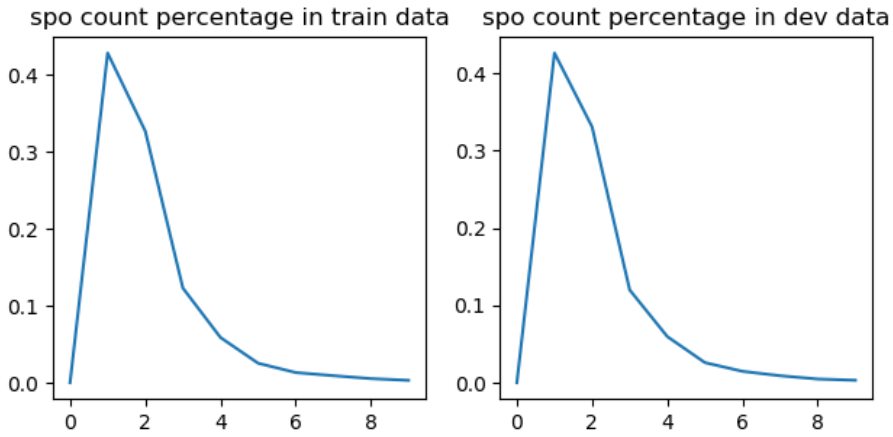

Figure 1: 三元组数量分布图

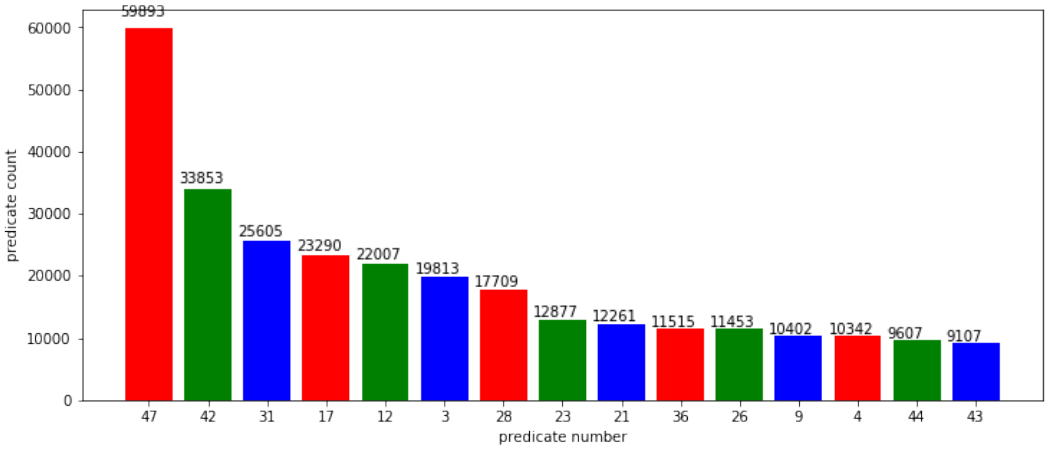

Figure 2: 训练数据中关系模式的出现次数统计

文本属性分析 使用 jieba 对训练集中的文本数据进行分词，统计各个词出现的次数，并生成对应的 wordcloud，如 Figure 3所示。可以发现，出现最多的词语是图书、作者、出版社，对应了上文中出现次数很多的人物-作者-图书作品关系模式，而演唱、一首、歌曲也对应了上文中出现次数很多的人物-歌手-歌曲关系模式。查看出现次数多的词语发现，这些词语都是生活化的词语，因此数据集是从日常生活中提取的。

2.2 数据处理

数据处理过程首先分别读取训练集、开发集和测试集文件数据，解析 json 字符串并提取出其中的数据，然后利用训练集数据构建词汇表，再使用词汇表编码训练集、开发集和测试集，接着用 pad 对数据长度进行补齐，然后用补齐的数据构建 pytorch 的 dataset，再进一步生成对应的 dataloader，最后用 pickle 导出训练集、开发集和测试集的 dataloader 以及过程中生成的词汇表。另外，词汇表使用了 fastNLP[2] 中的 Vocabulary，使用起来很方便。

2.2.1 普通多分类模型数据处理

CNN、RNN、BiLSTM、BiLSTM-pooling、RCNN 的多分类模型所需的数据处理过程如下，过程中出现的参数可见 Table 2。

[2] https://github.com/fastnlp/fastNLP

Figure 3: 训练数据的词云分析

1. 读取数据 json 文件，把每一行 json 字符串解析为一个字典，并取出每个`text` 以及对应`spo_list` 中的 predicate，如果是测试数据（没有 predicate），则用全 0 的 multi-hot 向量代替 predicate。
2. 把每一个 text 对应的一系列 predicate 转换为 multi-hot 向量，长度为 50。
3. 把 text 字符化，只使用训练数据构造字符 Vocabulary。另外，还可以使用数据中分的词来构造。此外，还可以使用 jieba 对 text 进行分词，然后构造对应的词 vocabulary。
4. 使用上述训练集的字符数据、词语数据分别生成字符级别、词语级别的 word2vec。
5. 使用上述 Vocabulary 编码全部数据，并在 text 开头补`<pad>` 对应的编码，把全部 text 都补全为统一长度`seq_len`。其中`seq_len` 比最长长度要长，取 320。
6. 用数据构造 pytorch 的 dataset，并生成对应的 dataloader。
7. 用 pickle 导出 dataloader 和 Vocabulary。

Table 2: 多分类模型数据处理参数

char vocab size	word vocab size	jieba word vocab size	max len	seq len	batch_size
8180	379700	246045	300	320	64

2.2.2 BERT 多分类模型数据处理

BERT 分类模型的输入数据处理过程如下。

1. 读取数据 json 文件，取出每个`text` 以及对应`spo_list` 中的 predicate，如果是测试数据，则用全 0 的 multi-hot 向量代替。
2. 使用 BertTokenizer 处理 text，获得 tokens。
3. 若 tokens 的长度大于`seq_len - 2`，则截取长度为`seq_len - 2`。
4. 在 tokens 前面添上`[CLS]`，在后面添上`[SEP]`。
5. 新建和 tokens 一样长的 0 向量，命名为`segment_ids`。
6. 使用 BertTokenizer 把 tokens 转换为 id，命名为`input_ids`。
7. 新建和`input_ids` 一样长的 1 向量，命名为`input_mask`。
8. 把`input_ids`、`input_mask` 和`segment_ids` 都用 0 补长为`seq_len`。
9. 为了命名一致，把`input_mask` 改为`attention_mask`，`segment_ids` 改为`token_type_ids`。

10. 把input_ids、attention_mask 和token_type_ids 作为模型 input，predicate 的 multi-hot 向量作为模型 target，构造 dataset，并生成对应的 dataloder，用 pickle 导出 dataloader 和 Vocabulary。其中，dataloader 的 batch size 设置为 16。

2.2.3 序列标注模型数据处理

以 BiLSTM、Transformer 作为编码器的分类模型的输入数据处理过程如下，数据处理过程中的参数可见 Table 3。

1. 读取数据 json 文件，读取出每个text 以及对应的分词序列、词性序列、SPO 序列。训练数据、开发数据中分别有 50、10 条数据是没有分词序列、词性序列的，因为数量少，直接忽略。由于训练的是标注模型，是需要给定关系类型的，则对于没有 SPO 的数据也同样忽略掉（对于没有关系类型的句子，在最后补上空的spo_list 即可）。
2. 序列化 text，命名为 char。text 中每个 char 还需要配上对应的词、词性信息，分别命名为 word、pos。检查三者的长度是否一致。
3. 每个 text 和对应的一种关系会有一种序列标注，实验中实现了 BIESO 和 BIEO 的标注方式，为了更好的标注效果，采用了 BIESO 的标注方式。标注时使用 text 以及对应一种关系的全部 SPO，利用正则表达式找到各个 subject 和 object 在 text 中的开始和结束位置。如果目标长度为 1 则标注为 S（如果是 BIEO 标注则标注为 B），长度为 2 则标注为 B 和 E，长度大于 2 的则 B 开头、I 中间、E 结尾。如果是 subject，则标注为 SUB，如果是 object 则标注为 OBJ，和 BIE 用横杠连接（如'B-SUB'）。标注序列命名为 tag。
4. 对于每个 text 的每种关系，都要生成一条数据 char, word, pos, spo, tag，构造 DataSet，其中 pos 为词性序列、spo 为关系对应的 one-hot 向量。对于测试数据，标注 tag 为全'O'。
5. 从训练数据中获取字的词典 char vocab、分词的词典 word vocab、词性的词典 pos vocab、标注符号的词典 tag vocab。
6. 利用上述词典来编码全部数据。
7. 设置char、word、pos 和spo 为 input，另外还需要生成 text 的长度为seq_len 序列，也作为模型 input，把序列标注tag 作为 target，构造对应的 dataset，并生成 dataloader。用 pickle 导出 dataloader 和各个 Vocabulary。

Table 3: 序列标注模型数据参数

train size	dev size	test size	char vocab	word vocab	pos vocab	label num	max len	seq len
303394	37977	17511	8163	379700	24	9	300	320

3 Method

3.1 整体架构

系统整体框架为把基于关系模式约束的主体-关系-客体信息抽取分割为串行的两个子任务：文本的关系模式多分类任务、给定关系模式的序列标注任务。其中，分类模型实现了 CNN、RNN、LSTM、LSTM pooling、RCNN、BERT 等模型，并在模型最后加一个线性层实现多分类功能。序列标注模型使用 BiLSTM、Transformer 作为 Encoder，结合 CRF 作为 Decoder，生成最佳标注序列。整体流程如下所示：

1. 数据预处理：读取数据，去除残缺数据项，构建 DataSet 和 Vocabulary。
2. 多分类：输入 text，输出 text 中符合 schema 约束的全部关系。
3. 序列标注：输入 text 和对应的一种关系，输出在该关系下的 text 的序列标注。
4. 提取三元组：从序列标注中提取出对应关系下的 subject 和 object，对主体和客体进行全连接构造三元组。
5. 输出数据：把对应同一个 text 的不同关系的 SPO 三元组组合到同一个spo_list 中，补上关系对应的subject_type 和object_type，输出最终结果。

3.2 Word2Vec

可以在训练数据的字符数据（char2vec）或者词语数据（word2vec）上使用 gensim 的 Word2Vec 来获得 embedding 的 weight，可能可以使 embedding 更高效，而可能使模型结果更好。

3.3 多分类模型

上文数据统计以及发现，文本所对应的关系种类并不惟一，绝大部分文本有 1 ~ 4 种关系，因此所需要的分类模型是多分类模型，模型要能输出每个类别的判别概率，如果概率大于 0.5，则认为该关系存在。

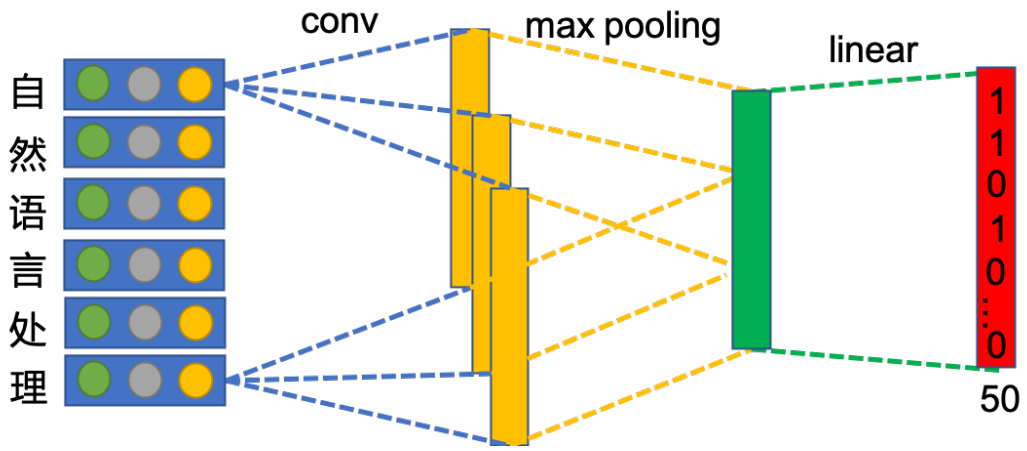

Figure 4: CNN 分类模型结构

CNN 模型结构如 Figure 4所示，输入的文本 text 先过一个 Embedding 层，再过一个卷积层，接着用 ReLU 激活，然后 max pooling，再拼接结果，然后 dropout，最后过全连接层并输出。

BiRNN 输入的文本 text 先过 1 个 Embedding 层，再进入双向 RNN，最后进入全连接层并输出。

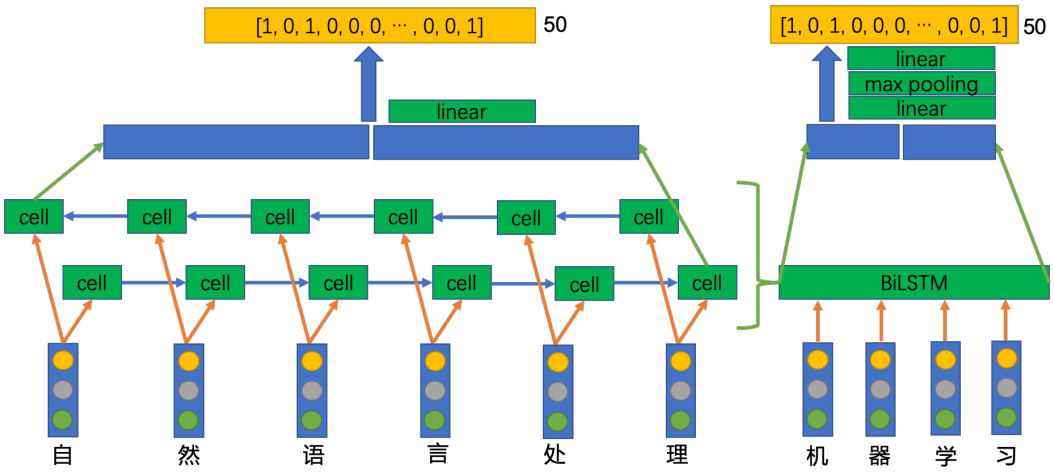

Figure 5: LSTM 分类模型结构与 RCNN 分类模型结构

BiLSTM 模型结构如 Figure 5左边部分所示，输入的文本 text 先过 1 个 Embedding 层，再进入双向 LSTM，然后 dropout，最后进全连接层并输出。

Bidirectional LSTM with max pooling 模型结构与 LSTM 类似，Zhou et al. [2016] 提出在进入全连接层之前先 max pooling，能有效提高文本分类效果。

RCNN 模型结构如 Figure 5右边部分所示，输入的文本 text 先过 1 个 Embedding 层，再进入双向 LSTM，之后进入线性层，然后 max pooling，最后进全连接层并输出。

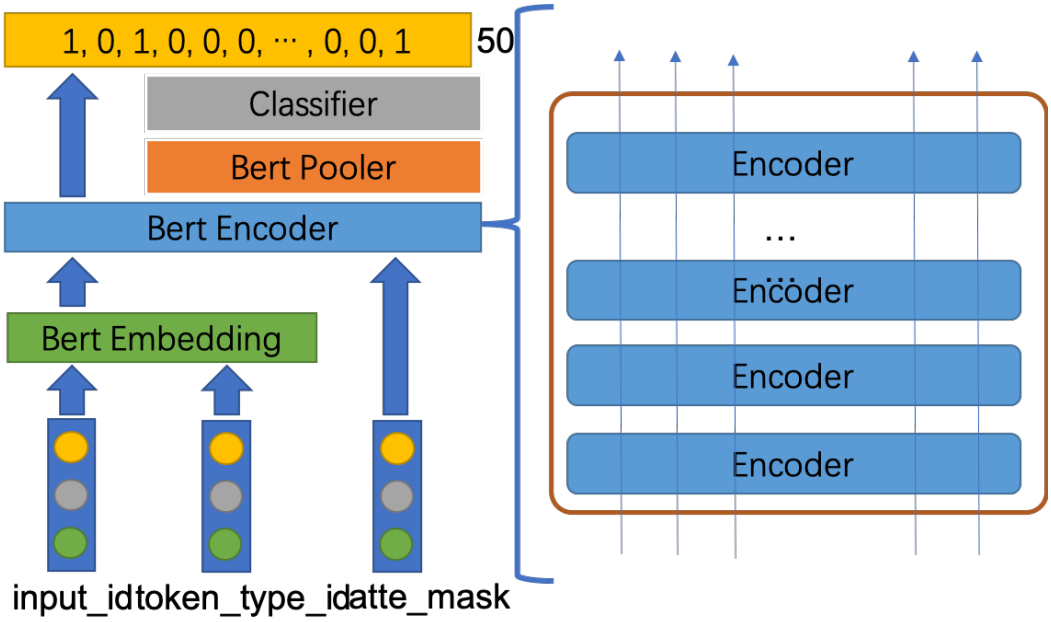

Figure 6: Bert 分类模型结构

BERT Devlin et al. [2018] 提出了 Bidirectional Encoder Representations from Transformers(BERT) 模型，结构如 Figure 6所示。BERT 由 Transformer Encoder（如 Figure 7右边部分所示）堆叠而成，使用 Masked LM 和 Next Sentence Prediction 来提炼字级别的表示，而且有在大数据集上 pretrain 的优势，能应用到分类模型中并达到不错的效果。

3.4 序列标注模型

序列标注模型采用了标准的 Encoder、Decoder 结构。序列标注模型框架请看 Figure 7左边部分。其中，BiLSTM 以及 Transformer 都是很好的 Encoder 的选择，而 Decoder 则采用了 Conditional Random Field(CRF)。

特征构造 在数据进入 Encoder 前，需要先获得 Embedding。为了能够充分利用数据集中提供的字、词、词性信息，把字、词、词性三者的 Embedding 拼接在一起，最后还要拼接上关系类别的 one-hot 表示，作为最终输入 Encoder 的 Embedding。Embedding 结构如 Figure 8所示。其中，词的部分可以采用数据原有的分词，如 Figure 8上半部分所示，但是如上文所述，数据中给出的分词并不彻底，因此也可以采用 jieba 的分词结果，如 Figure 8下半部分所示。

Encoder: BiLSTM BiLSTM 结构如 Figure 5所示，其结构在上文已经叙述，故不再重复。

Encoder: Transformer Vaswani et al. [2017] 提出了 Transformer 模型。Transformer 抛弃了传统的 LSTM 结构，仅由 self-Attenion 和 Feed Forward NeuralNetwork 组成，可以有效地学习文本内容。本次实验可以使用 Transformer 的 Encoder 部分，作为序列标注的 Encoder。Transformer Encoder 结构请看 Figure 7右边部分。

Decoder: CRF 在序列标注的时候，如果直接取每一个位置分数最高的标注，这样的结果往往是不好的，因为序列标注存在着很强的前后关联性，例如 I 后面必须是同一类标注的 I 或者 E。因此需要给定一个特征矩阵以及转移分数矩阵，可以计算出序列标注的最佳的路

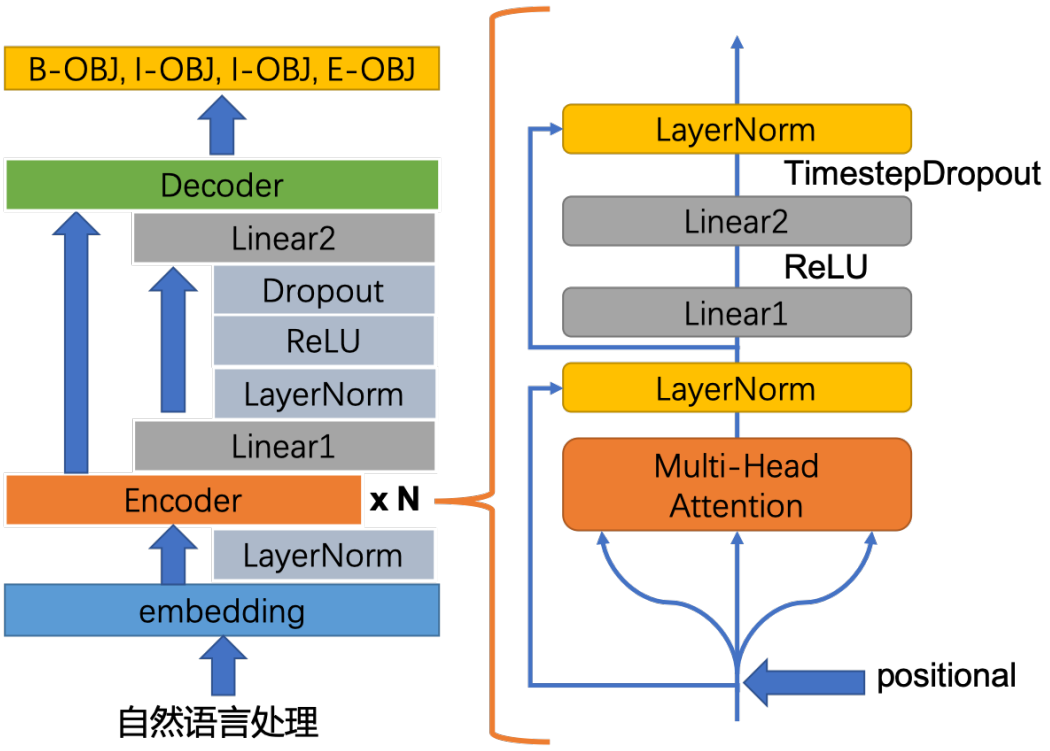

Figure 7: 序列标注模型结构与 Transformer 编码器结构

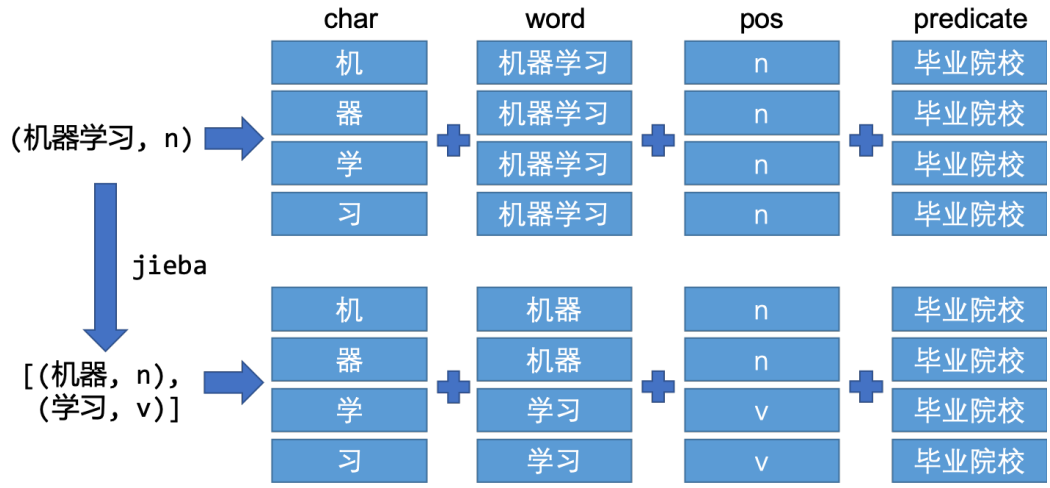

Figure 8: 序列标注模型的输入特征构造

径以及对应的分数。我实现了基于 BIESO 标注方式的转移规则，如 Table 4所示。由于采用的是 BIESO 标注方式，而需要标注的对象有主体 SUB 和客体 OBJ，因此全部的标注有 B-SUB,I-SUB,E-SUB,S-SUB,B-OBJ,I-OBJ,E-OBJ,S-OBJ,O 一共 9 种标注。

Table 4: BIESO 序列标注类型的转移矩阵

	start	B	I	E	S	O	end
start	N	Y	N	N	Y	Y	N
B	N	N	-	-	N	N	N
I	N	N	-	-	N	N	N
E	N	Y	N	N	Y	Y	Y
S	N	Y	N	N	Y	Y	Y
O	N	Y	N	N	Y	Y	Y
end	N	N	N	N	N	N	N

4 Experiment

4.1 模型参数

CNN 模型参数如 Table 5所示，RNN 类模型（包括 RNN、LSTM、LSTM pooling 和 RCNN）参数如 Tabel 6所示。由于 BERT 模型是从中文 pretrained`chinese_L-12_H-768_A-12` 模型[3] finetune 而来，因此参数设置都是官方设置，故在此不再叙述。而序列标注模型的 embedding 参数设置请看 Table 7，而以 BiLSTM 作为 Encoder 的序列标注模型参数请看 Table 8，而以 Transformer 为 Encoder 的序列标注模型参数请看 Table 9。

Table 5: CNN 分类模型参数

class num	embed dim	kernel size	kernel num	in channels	dropout
50	128	(3,4,5)	128	1	0.5

Table 6: RNN 分类模型参数

model	layers	bidirectional	embed dim	hidden dim	output dim	pooling	linear layers	dropout
RNN	1	T	128	256	50	F	1	0.5
LSTM	1	T	128	256	50	F	1	0.5
LSTMmxp	1	T	128	256	50	T	1	0.5
RCNN	1	T	128	256	50	T	2	0.5

4.2 模型训练

1. epoch 上限：除了 BERT 模型的 epoch 上限为 3，其他模型统一设置为 64。
2. early stop：全部模型均使用了 early stop，patience 为 10。
3. timing：全部模型都使用了 time 进行计时。
4. metric：分类模型的评测标准使用 F1 值，通过向量运算可以获得 F1、precision 和 recall 值。序列标注模型的评测使用了 fastNLP 的 SpanFPreRecMetric，计算序列标注的 F1 值。
5. batch size：由于 BERT 模型训练占用 GPU 内存空间大，设置 batch size 为 16，其他模型的 batch size 均为 64。
6. optimizer：各个模型对应的 optimizer 设置请看 Table 10。
7. loss：分类模型均使用 pytorch 的 binary_cross_entropy_with_logits() 现。序列标注模型由于使用了 CRF 作为 Decoder，需要在模型内部计算 Negative Log Likelihood Loss。
8. tensorboard：模型训练都使用了 tensorboardx 来记录模型训练过程中的 loss、F1、precision、recall 等值。其中，CNN 模型训练的 tensorboard 记录如 Figure 9所示。

[3] https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip

Table 7: 序列标注模型 Embedding 设置

char embed	word embed	pos embed	spo dim	total size
64	64	64	50	242

Table 8: LSTM 序列标注模型参数

	RNN	LSTM	LSTM pooling	RCNN
layers	1	1	1	1
bidirectional	T	T	T	T
embed dim	256	256	256	256
output dim	50	50	50	50
pooling	F	F	T	T
linear layers	1	1	2	2

5 Results

5.1 分类模型结果与对比

分类模型结果请看 Table 11。其中 BERT 的整体分类效果最好，F1 值达到了 90.39%。除了 RNN 的效果很差以外，其他的模型如 CNN、LSTM、RCNN 效果都差不多。另外，字符级别的 char2vec 和 jieba 分词的 word2vec 并没有什么效果。

5.2 序列标注模型结果与对比

序列标注结果请看 Table 12。其中以 LSTM 为 Encoder 的标注效果更好，F1 值达到 90.82%。原本估计 Transformer 的标注效果会更好，因为 Transformer 融入了 attention 的机制，但是最后结果却是 LSTM 的更好。其中一个原因可能是参数设置的问题。实验测试了不同的 inner size、key size、value size 以及 linear 层的数量对序列标注结果的影响，发现只有 1 层线性层、inner size 为 256、key size 和 value size 为 64 时达到最好的效果，F1 值为 85.14%。经对比发现，1 层线性层的效果会更好。详细对比结果请看 Table 13。另外，jieba 分词的 word2vec 并没有什么效果。

6 Discussion

本次项目实现了从句子中提取出句子中的关系模式及其对应主体和客体的系统。系统把提取过程分为两步：多分类和序列标注。项目在 SKE 上，训练了多个多分类模型，包括 CNN、RNN、BiLSTM、BiLSTM-pooling、RCNN 的简单分类模型，以及 BERT；而且，还分别实现了使用 BiLSTM、Transformer 作为编码器的序列标注模型。其中，最好的多分类模型是 BERT，在开发集上的 F1 值达到 90.30%，而最好的序列标注模型是使用 BiLSTM 作为编码器的，其 F1 值达到了 85.14%。因此，总体上来说，系统可以较好地从句子中提取出含有的关系以及对应的主体和客体。

但是项目也有一些做的不足的地方。没有实现模型的 ensemble。此外，还可以考虑用 bert 来做序列标注。而且最后评价模型的指标不应该用在开发集上的结果来衡量模型好坏，应该在测试集上测出的指标才能衡量模型的能力，但是由于测试集没有给出 spo_list，所以没办法测出结果，所以应该一开始的时候就从开发集中划出一小部分作为测试集，最后在这一小部分测试集上测出模型能力。

Table 9: Transformer 序列标注模型参数

layers	inner size	key size	value size	num head	class num	dropout
4	256	64	64	4	9	0.1

Table 10: Optimizer 参数

model	optim	lr	weight decay	warmup proportion
CNN 等分类模型	Adam	1e-3	0	-
BERT 分类模型	BertAdam	5e-5	1e-2	0.1
序列标注模型	Adam	1e-3	0	-

Figure 9: CNN 分类模型训练的 tensorboard 记录

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Table 11: 分类模型训练结果

model	embedding	F1	recall	precision	best epoch
CNN	-	85.91	81.6	**90.69**	17
	char2vec	85.22	80.91	90.01	18
	word2vec	85.3	80.45	89.39	13
RNN	-	60.73	49.33	78.97	4
LSTM	-	87.76	86.08	89.51	33
LSTM pooling	-	88.67	87.29	90.11	64
	char2vec	86.54	84.3	88.89	53
RCNN	-	87.62	86.38	88.9	4
	char2vec	87.62	85.89	89.71	4
BERT	-	**90.3**	**89.96**	90.64	3

Table 12: 序列标注模型训练结果

encoder	embedding	F1	recall	precision	best epoch
BiLSTM CRF	-	**90.82**	**90.77**	**90.88**	5
	word2vec	89.71	90.00	89.42	4
Transformer CRF	-	85.14	85.16	85.12	9
	word2vec	79.85	79.75	79.96	14

Table 13: Transformer 模型训练结果对比

inner size	key size	value size	linear layers	F1	recall	precision	best epoch
128	32	32	1	0.842039	**0.863064**	0.822014	7
128	64	64	1	0.841277	0.824173	**0.859105**	6
256	32	32	1	0.8415	0.844386	0.838634	7
256	64	64	1	**0.85137**	0.851564	0.851175	9
256	64	64	2	0.521882	0.674875	0.425436	4
256	128	128	2	0.498182	0.624558	0.414342	2

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.

A 代码组织

如果需要查看代码，请看 Table 14了解代码组织的结构。

Table 14: 代码组织

文件夹名称	内容
`classification`	除了 BERT 以外的多分类模型代码
`classification-bert`	BERT 多分类模型代码
`labeling`	序列标注模型代码
`analysis`	数据分析代码