声明：我已知悉学校对于考试纪律的严肃规定，将秉持诚实守信宗旨，严守考试纪律，不作弊，不剽窃；若有违反学校考试纪律的行为，自愿接受学校严肃处理。

2018-2019 学年第二学期 COMP130137.01 《模式识别与机器学习》课程项目 基于 schema 约束的 SPO 信息抽取任务

学号：16307130194 , 姓名：陈中钰，贡献: 100%，签名：

Abstract

本次实验项目把基于 schema 约束的 SPO 信息抽取任务分割为串行的两个子任务：（1）文本的 schema 关系分类任务和（2）给定 schema 关系的序列标注任务。实验首先实现了基于 CNN、LSTM、RCNN 和 BERT 的分类模型，在开发集上达到 F1 score 为 90.30%d。然后实验实现了结合 CRF 的 BiLSTM、Transformer 的序列标注模型，在开发集上达到 F1 score 为 85.14%。实验最后对分类模型和序列标注模型分别实现了简单的 ensemble，并从最终的序列标注结果中提取出文本对应的 SPO 关系。实验全过程使用了 fastNLP，包括数据集处理、模型搭建、模型训练、模型测试等步骤，并对 fastNLP 做了一些改进。

1 背景

1.1 问题描述

任务给定一系列schema 约束，每个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。要求输入句子文本text、句子分词和对应词性postag，输出所有满足给定schema 约束的SPO 三元组知识Triples=[(S1，P1，O1)，(S2，P2，O2)...]。

1. 输入样例
   (a) schema 约束,如{subject_type:影视作品，predicate:改编自，object_type:作品}
   (b) text= “《端脑》改编自有妖气同名漫画《端脑》”
   (c) postag=[{"word": "《", "pos": "w"}, ...]
2. 输出样例
   spo_list=[{"predicate": "改编自", "object_type": "作品",
   "subject_type": "影视作品", "object": "端脑", "subject": "端脑"}, ...]
3. 评价指标
   评价指标为F1值，precision以及recall，其中F1值为第一关键字。

1.2 技术框架

主要解决方法为把基于 schema 约束的 SPO 信息抽取任务分割为串行的两个子任务：（1）文本的 schema 关系分类任务和（2）给定 schema 关系的序列标注任务。

1. 分类模型：基于 CNN、LSTM、RCNN、Transformer、BERT 等模型，最后加一个线性层实现分类功能。

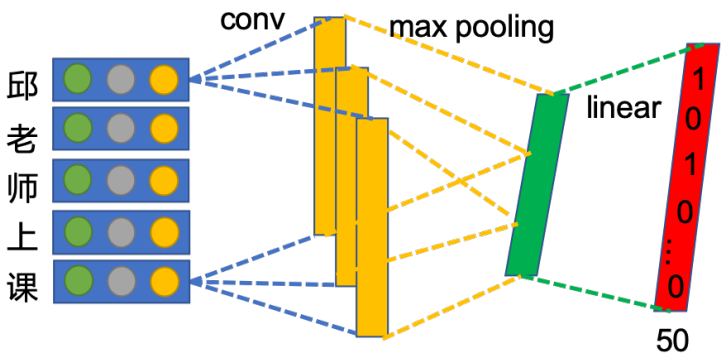

Figure 1: CNN 模型结构

2. 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。

2 实验方法

2.1 分类模型

通过对比赛数据统计可以发现，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。

2.1.1 CNN

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。CNN 模型结构请看 Figure 1。

2.1.2 Bidirectional RNN

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。

2.1.3 Bidirectional LSTM

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。LSTM 模型结构请看 Figure 2。

2.1.4 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。

2.1.5 RCNN

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。RCNN 模型结构请看 Figure 2。

2.1.6 BERT

Devlin et al. [2018] 提出了 Bidirectional Encoder Representations from Transformers(BERT) 模型，由 Transformer Encoder 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。BERT 模型结构请看 Figure 3。

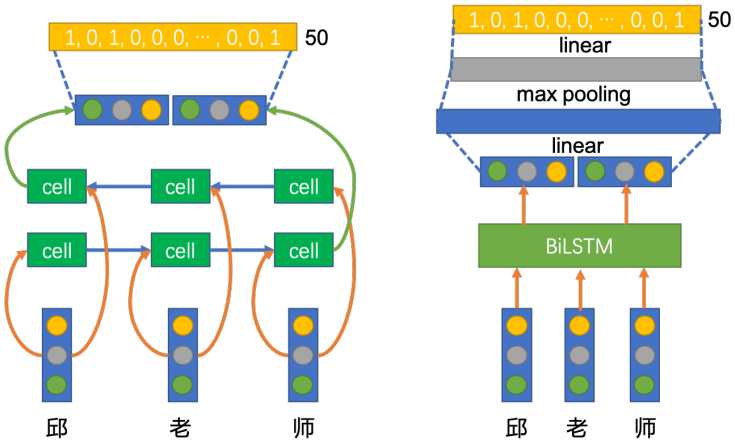

Figure 2: （左）BiLSTM 模型结构，（右）RCNN 模型结构

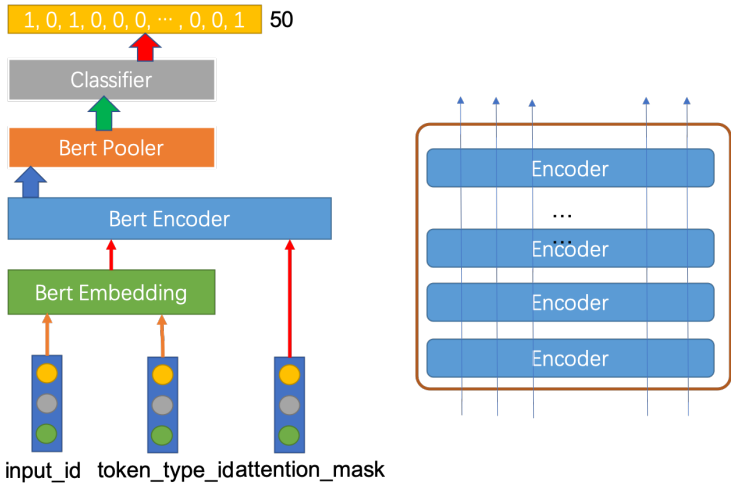

Figure 3: （左）BERT 模型结构，（右）BERT Encoder 结构

2.2 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 4。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 结构请看 Figure 5。

2.2.1 Encoder

Bidirectional LSTM 以及 Transformer 都是很好的 Encoder 的选择。
BiLSTM 结构在上文已经叙述，故不再重复。
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，在作为序列标注的 Encoder。Transformer Encoder 结构请看 Figure 6。

2.2.2 Decoder: Conditional Random Field

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

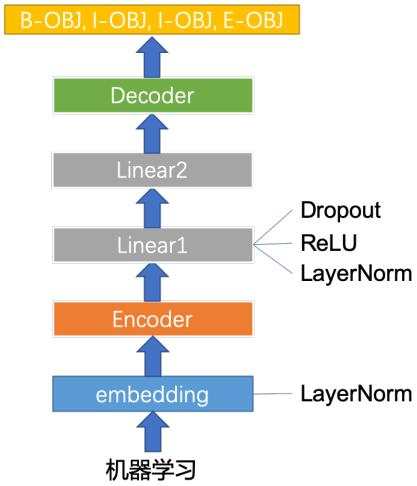

Figure 4: 序列标注模型框架

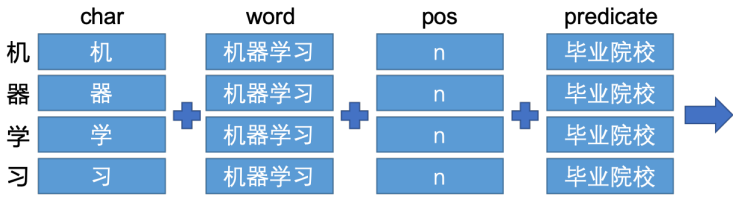

Figure 5: 序列标注模型的 Embedding 构造

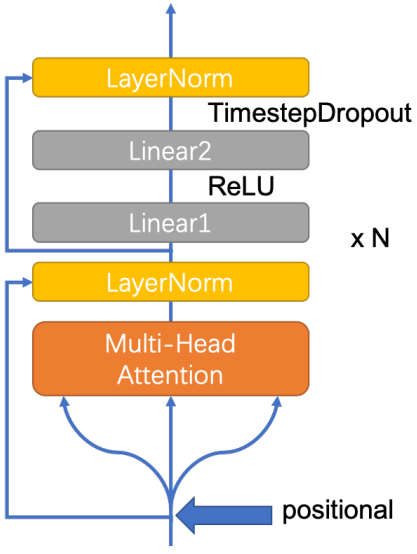

Figure 6: Transformer Encoder 结构

Table 1: 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

Table 2: 分类模型数据参数

train size	dev size	test size	vocab size	class num	max len/seq len
173108	21639	9949	8163	50	300/320

以及转移分数矩阵，可以计算出序列标注的最佳的路径以及对应的分数。
在原来 fastNLP 的基础上，我添加了 BIESO 和 BIEO 两种标注方式的转移规则（BIESO 的转移规则请看 Table 1），并最终使用了 BIESO 的标注方式。

3 实验设计

3.1 实验框架

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 SPO 三元组：给定 text 和关系，从序列标注中提取出该关系下的 subject 和 object，构造 SPO 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 数据处理

数据处理主要基于 fastNLP 的 DataSet 和 Vocabulary 实现。首先读取文件数据并构造 DataSet，再通过对 DataSet 进行操作来构造 Vocabulary，并对 DataSet 里的数据进行编码，最后使用 pickle 导出处理好的 DataSet 和 Vocabulary。

3.2.1 分类模型数据处理

1. 读取数据 json 文件，把每一行解析为一个字典，并取出每个`text` 以及对应`spo_list` 中的 predicate，构造 DataSet。
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 序列化，只使用训练数据构造 Vocabulary。
4. 使用上述 Vocabulary 编码全部数据，并在 text 开头补`<pad>` 对应的码，把全部 text 都补全为统一长度`seq_len`。
5. 设置输入为 text、输出为 multi-hot 向量，用 pickle 导出 DataSet 和 Vocabulary。

3.2.2 BERT 分类模型数据处理

BERT 分类模型的输入数据同样使用 fastNLP 的 DataSet 和 Vocabulary 完成。

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，构造 DataSet。

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

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. 设置`input_ids`、`attention_mask` 和`token_type_ids` 为 input，predicate 的 multi-hot 向量为 target，用 pickle 导出 DataSet 和 Vocabulary。

3.2.3 序列标注模型数据处理

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 input。设置序列标注`tag` 为 target。用 pickle 导出 DataSet 和各个 Vocabulary。

3.3 模型参数

CNN 模型参数、RNN 类模型（包括 RNN、LSTM、LSTM with maxpooling 和 RCNN）参数请见 Tabel 4 和 5。由于 BERT 模型是从中文 pretrained`chinese_L-12_H-768_A-12` 模型 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 设置请看 Table 6，以 LSTM 作为 Encoder 的模型参数请看 Table 7，以 Transformer 为 Encoder 的模型参数请看 Table 8。

Table 4: 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 5: 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

3.4 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 Timing 计时 Callback。
4. metric：分类模型的评测标准使用了自己编写的 F1 类，通过向量运算可以获得 F1、precision 和 recall 值，并设置 F1 值为评测值。序列标注模型的评测标准使用了 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 9。
7. loss：分类模型均使用自己实现的 BCEwithLogitsLoss，基于 pytorch 的 binary cross entropy with logits 实现。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. fitlog：模型训练都使用了 fitlog 的`add_other` 来记录参数设置，并且使用 fitlog callback 记录每个 epoch 的评测结果。
9. trainer：全部模型都使用了 Trainer 类来训练模型。

3.5 模型测试

模型测试通过 Tester 类实现。

3.6 模型 ensemble

在 ensemble 之前，首先导入数据，并用 Tester 检测 F1 值，判断是否导入了正确的模型。ensemble 采用了普通的投票机制：对输出结果的概率进行加权平均，作为 ensemble 的结果。

3.7 模型 predict

使用 ensemble 的模型权重分配来获得最终结果，并把最终概率转换为目标输出。

Table 6: 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 7: LSTM 序列标注模型参数

layers	bidirectional	embed dim	hidden dim	class num	linear layers	dropout
2	T	242	256	9	2	0.5

Table 8: 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

4 结果分析

4.1 分类模型结果与对比

分类模型结果请看 Table 10。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。

4.2 序列标注模型结果与对比

序列标注结果请看 Table 11。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 12。

5 结论

本次实验项目把基于 schema 约束的 SPO 信息抽取任务分割为串行的两个子任务：（1）文本的 schema 关系分类任务和（2）给定 schema 关系的序列标注任务。实现了基于 CNN、LSTM、RCNN 和 BERT 的分类模型，其中 BERT 具有最高的 F1 值，在开发集上达到了 90.30%d。接着实现了以 BiLSTM、Transformer 为 Encoder、CRF 为 Decoder 的序列标注模型，其中以 BiLSTM 为 Encoder 的 F1 值更高，在开发集上达到为 85.14%。最后还对分类模型和序列标注模型分别实现了简单的 ensemble。
预测过程先生成关系结果，再根据关系结果生成的序列标注，最后从序列标注中提取出文本对应的 SPO 关系。
实验全过程使用了 fastNLP，包括数据集处理、模型搭建、模型训练、模型测试等步骤，并对 fastNLP 做了一些改进。

6 代码组织

如果需要查看代码，请看 Table 13 了解代码组织的结构。

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.

Table 9: 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	-

Table 10: 分类模型训练结果

model	F1	recall	precision	best epoch
CNN	0.859051	0.815966	**0.906939**	17
RNN	0.607297	0.493328	0.7897449	4
LSTM	0.87761	0.860816	0.895071	33
LSTMmxp	0.886762	0.872898	0.901073	64
RCNN	0.876248	0.863817	0.889042	4
BERT	**0.903014**	**0.899639**	0.906415	3

Table 11: 分类模型训练结果

encoder	F1	recall	precision	best epoch
LSTM	**0.90824**	**0.907684**	**0.908798**	5
Transformer	0.85137	0.851564	0.851175	9

Table 12: 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

Table 13: 代码组织

filename	function
`chinese_L-12_H-768_A-12`	预训练的 BERT Chinese 模型
`convert_tf_checkpoint_to_pytorch.py`	转换 BERT 为 pytorch 的 checkpoint
`tf2pytorch_ckpt.sh`	转换 BERT 为 pytorch 的 checkpoint
data	存放比赛数据
`classification\config.py`	定义了全部参数的 Config 类
`classification\dataset.py.py`	定义了分类模型的数据处理
`classification\dataset_bert.py`	定义了 BERT 分类模型的数据处
`classification\model.py`	定义了分类模型
`classification\model_bert.py`	定义了 BERT 分类模型
`classification\train.py`	定义了分类模型的训练过程
`classification\train_bert.py`	定义了 BERT 分类模型的训练过程
`classification\ensemble.py`	实现了简单的 ensemble
`classification\predict.py`	生成关系预测文件
`classification\utils.py`	定义了文件读取函数、Loss 类、Callback 类和 Optimizer 类
`labeling\config.py`	定义了全部参数的 Config 类
`labeling\dataset.py.py`	定义了序列标注模型的数据处理
`labeling\tagging.py`	实现了 BIESO 和 BIEO 的序列标注函数
`labeling\model.py`	定义了序列标注模型型
`labeling\crf.py`	添加了 BIESO 和 BIEO 的转换模式
`labeling\train.py`	定义了序列标注模型的训练过程
`labeling\ensemble.py`	实现了简单的 ensemble
`labeling\predict.py`	生成 SPO 预测文件
`labeling\utils.py`	定义了文件读取函数、Callback 类

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.

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.