目录

（一）背景及意义

1.1 背景

随着计算机的应用普及，人们逐渐考虑用计算机进行文本处理。而现阶段的文本主要是由人们使用自然语言写成的，那么文本中就不可避免的包含了作者的感情。为了让计算机能更好地理解文本的内容，就需要对文本的情感进行分析。

在情感原因提取任务提出之前，文本情感分析相关的研究主要集中在情感分类任务和情感要素提取任务上。经过多年的发展，与这些任务相关的技术已经较为成熟。后来，EMNLP 2017 年的论文《A Question Answering Approach for Emotion Cause Extraction》[1]提出了对文本中情感的原因进行提取，也就是情感原因提取任务。

文本情感原因提取主要研究从文本中自动识别导致情感产生和变化的因素或事件的方法。对于包含情感的文本，在情感分析研究解决了对文本中的情感“知其然”的基础上，情感原因提取的研究尝试更进一步发现情感产生和变化的原因，也就是“知其所以然”。

1.2 意义

文本情感原因提取任务的发展，有利于帮助解决自然语言处理领域语义理解、语用分析等方向的难题。另外，随着互联网的飞速发展，社交媒体日益成为人们日常生活中不可或缺的一部分。将情感原因提取应用于社交平台，有利于更好的了解用户情感状态变化，也有利于及早的发现用户情感状态方面的异常，进而避免悲剧发生。

（二）数据集（训练、测试、抑郁数据采集）

2.1 新浪新闻语料库（总计 2105 个实例）

以 3 年（2013-2015 年）的时间从 NEWS SINA2（共 20,000 条文章）中获取的中国城市新闻作为原始语料[2]。基于 10259 个中文主要情感关键词（简称关键词）的列表（Xu 等，2008 年），通过关键字匹配从原始数据中提取了 15687 个实例。在这里，称情感关键词的存在为语料库中的一个实例。对于每个匹配的关键字，提取三个前面的子句和三个下面的子句作为实例的上下文。如果一个句子在每个方向上都有 3 个以上的从句，则上下文将包括句子的其余部分以使上下文完整。为了简单起见，省略了跨段落上下文。

删除不相关的实例后，仍然还有 2,105 个实例。对于每个情感实例，两个注释者以 W3C 情感标记语言（EML）格式手动注释情感类别和原因。情感原因标记为<原因>，情感关键字标记为<关键词>。情绪类型，POS，位置和注释的长度也以 Emotionml 格式注释。

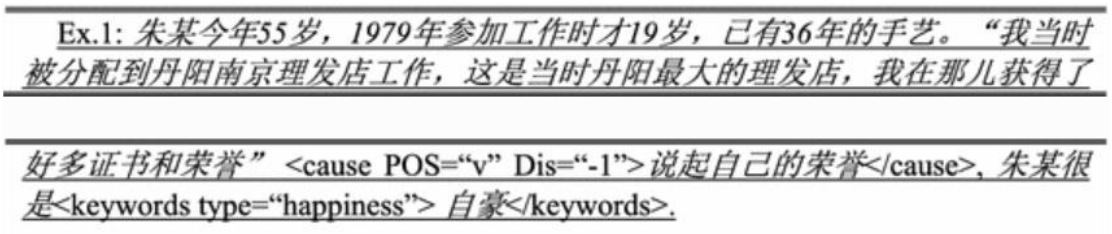

图 2.1：新浪新闻语料库示例

2.2 基于新浪新闻的 ECPE 语料库（总计 1945 份文档）

基于基准的 ECE 语料库构建的一个 ECPE 语料库，其中每个文档只包含一种情感和对应的一个或多个原因[3]。具有两种或两种以上情绪的文档被分成几个样本，以便每个样本只包含一种情绪。为了更好的满足 ECPE 任务设置，具有相同文本内容的文档被合并为一个文档，并在该文档中对每个情感原因对进行标记。在合并后的数据集中不同情感-原因对数量的文档所占比例如下表所示：

	Number	Percentage
Documents with one emotion-cause pair	1746	89.77%
Documents with two emotion-cause pairs	177	9.10%
Documents with more than two emotion-cause pairs	22	1.13%
All	1945	100%

表 2.1：情感-原因对文档比例

原语料库中的情感类别注释未被使用，训练过程只使用了情感子句和对应原因子句的位置信息。例如：

第 213 号文档中共有 10 个子句，情感子句为 2，对应的原因子句同样为 2，即情感-原因子句对为（2，2）；

第 12 号文档中共有 12 个子句，情感子句为 12，对应的原因子句有 9、10、11，即情感-原因子句对为（12，9）、（12，10）、（12，11）。

```
213 10
 (2,2)
1,null,null,扶个 老人 真的 是 土豪 才能 做 的 好事 吗
2,sadness,心寒,南京 彭宇案 后 多 起 扶 老 被 讹 案令 不少 路 人 遇事 心寒 选择 避走
3,null,null,然而
4,null,null,扶 老 就 一定 会 被 讹 吗
5,null,null,未必
6,null,null,此前 有 河源市 三名 法官 勇扶雨 中 摔倒 的 八旬 老太
7,null,null,最近 有 东莞 小伙 撞 人 肇事 被判赔 28 万多元
8,null,null,事实证明
9,null,null,老人 一口咬定 的 不 一定 能定 赔
10,null,null,抱 侥幸 心态 抵赖 的 也 推脱 不 掉 责任
```

图 2.2：数据集示例

```
4 12
 (12,9), (12,10), (12,11)
1,null,null,为 尽快 将 女子 救 下
2,null,null,指挥员 立即 制订 了 救援 方案
3,null,null,第一组 在 楼下 铺设 救生 气垫
4,null,null,并 对 周围 无关 人员 进行 疏散
5,null,null,另一组 队员 快速 爬 上 6 楼
6,null,null,在 楼 内 对 女子 进行 劝说
7,null,null,劝说 过程 中
8,null,null,消防官兵 了解 到
9,null,null,该 女子 是 由于 对方 拖欠 工程款
10,null,null,家中 又 急需 用钱
11,null,null,生活 压力 大
12,sadness,无奈,无奈 才 选择 跳楼 轻生
```

图 2.3：数据集示例

2.3 抑郁数据采集

我们使用爬虫技术从百度贴吧抑郁吧爬取用户言论数据，然后对数据进行分词处理，用于模型的训练和测试。

首先构建 url，调用 urllib 库函数获取网站代码，对页面标签进行处理后使用模式匹配得到需要的用户数据。

```python
def getContent(self, page):
    # 匹配所有楼层的内容
    pattern = re.compile('<div id="post_content_.*?>(.*?)</div>', re.S)
    items = re.findall(pattern, page)
    contents = []
    for item in items:
        # 将文本进行去除标签处理，同时在前后加入换行符
        content = self.tool.replace(item)
        contents.append(content.encode('utf-8'))
    return contents
```

图 2.4：爬虫代码

最后将网页中所有的用户言论数据写入文件保存，得到如下数据集：

content1.txt	2020/12/25 21:05	文本文档	5 KB
content2.txt	2020/12/25 19:53	文本文档	16 KB
content3.txt	2020/12/25 20:41	文本文档	50 KB
content4.txt	2020/12/25 20:49	文本文档	34 KB
content5.txt	2020/12/25 20:51	文本文档	13 KB
content6.txt	2020/12/19 11:04	文本文档	24 KB
content7.txt	2020/12/19 11:06	文本文档	26 KB
content8.txt	2020/12/19 11:07	文本文档	9 KB
content9.txt	2020/12/19 11:10	文本文档	33 KB
content10.txt	2020/12/19 13:46	文本文档	18 KB
content11.txt	2020/12/19 13:52	文本文档	10 KB
content12.txt	2020/12/19 13:54	文本文档	22 KB
content13.txt	2020/12/19 13:56	文本文档	10 KB
content14.txt	2020/12/19 13:58	文本文档	8 KB
content15.txt	2020/12/19 13:59	文本文档	19 KB
content16.txt	2020/12/19 14:02	文本文档	10 KB
content17.txt	2020/12/19 14:03	文本文档	20 KB
content18.txt	2020/12/19 14:04	文本文档	9 KB
content19.txt	2020/12/19 14:06	文本文档	30 KB
content20.txt	2020/12/19 14:08	文本文档	34 KB

图 2.5：爬取文本

得到数据集之后，对数据进行分词处理。使用 python 中的 jieba 库，调用库中的 cut 函数，使用精确模式，以数据集中的一句话为单位，对句中的每个子句进行分词处理，然后将分词后的一句话写入一个数据集中，这样最终得到 1200 多个分词数据集。

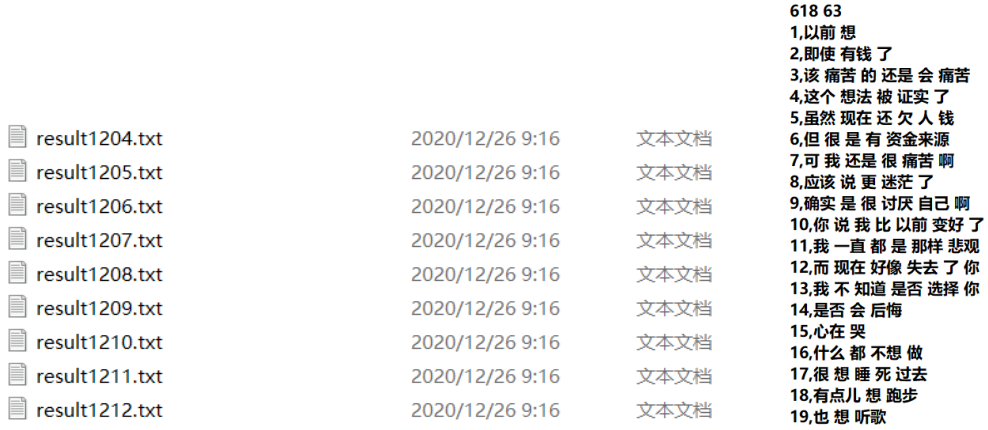

图 2.6：分词数据

（三）ECPE 任务模型介绍

参考论文**《Emotion-Cause Pair Extraction: A New Task to Emotion Analysis in Texts》**[3]，提出一个全新的情感原因对提取任务用来实现情感原因的提取。文中提出了一种两步方法。

第一步：将情感原因对的提取任务转换为两个独立的子任务(情感提取和原因提取)。使用多任务学习网络，判断文本中的每一个子句是情感子句的概率和是原因子句的概率，从而提取出情感子句集 E 和原因子句集 C。

第二步：情感原因对的配对和过滤。对 E 和 C 应用笛卡尔积运算，得到所以可能的情感原因对组成的集合。然后使用过滤器删除不包含因果关系的对。

针对步骤一，本文提出了独立多任务学习和交互式多任务学习两种解决办法，后者是前者的加强版本，在前者的基础上利用了情感和原因之间的关联。设文档 d 包含多个子句：$d=[c_1,c_2,\ldots,c_d]$，每个子句 c_i 包含多个单词：$c_i=[w_{i,1},w_{i,2},\ldots,w_{i,c}]$。

3.1 第一步：

- **独立多任务学习：**

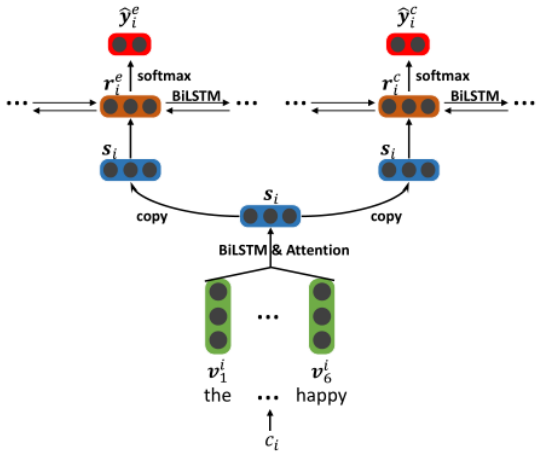

Figure 2: The Model for Independent Multi-task Learning (Indep).

图 3.1：情感原因提取模型架构

如图 3.1 所示，独立多任务学习 Indep 分上下两层 *BiLSTM* 。下层是单词级 *BiLSTM* ，子句 c_i 的每个单词输入后用 *BiLSTM* 和注意力机制获得子句的表示 s_i 。上层有两个子句级 *BiLSTM* ，一个负责情感提取，一个负责原因提取，两部分间没有数据共享。从下层获得的 s_i 输入上层的 *BiLSTM* 后，得到子句的 *contextaware* 特征，然后输入到 softmax 对子句进行分类。

- **交互式多任务学习：**

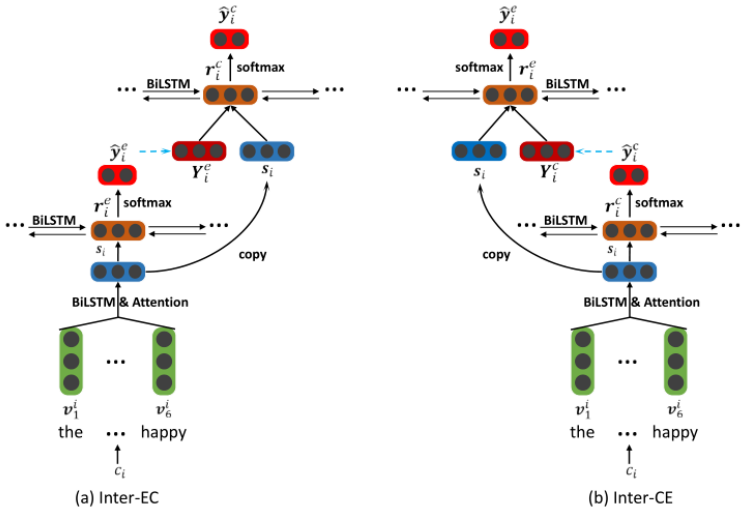

Figure 3: Two Models for Interactive Multi-task Learning: (a) Inter-EC, which uses emotion extraction to improve cause extraction (b) Inter-CE, which uses cause extraction to enhance emotion extraction.

图 3.2：情感原因配对模型架构

Indep 中上层的两个 *BiLSTM* 相互独立。在此基础上本文提出了交互式多任务学习，

一种是情感辅助原因($Inter-EC$)，另一种是原因辅助情感($Inter-CE$)。 以 $Inter-EC$ 为例来看，模型将得到的 y_i^e 嵌入到 Y_i^e 中，以 $y_i^e \oplus s_i$ 作为输入，经 $BiLSTM$ 和 softmax 得到原因提取结果。

3.2 第二步：

对情感子句集 E 和原因子句集 C 应用笛卡尔积运算，得到所有情感原因对组成的集合 P_{all} ：

$$P_{all} = \{\ldots,(c_i^e, c_j^c),\ldots\}$$

然后将 P_{all} 中的每个对转化为一个三维向量(s_i^e ， s_j^c 分别是两个子句的表示，v^d 是二者之间的距离)，

$$X_{(c_i^e, c_j^c)} = [s_i^e, s_j^c, v^d]$$

最后用 Logistic 回归检测每个候选对，以判断 c_i^e 和 c_j^c 之间是否存在因果关系(y=1 表示存在因果关系，否则不存在因果关系)。

$$Y_{(c_i^e, c_j^c)} \leftarrow \delta(\theta^T X_{(c_i^e, c_j^c)})$$

（四）基础模型的不足与改进

4.1 现有方法&改进方法

现有方法: 预训练的 word2vec

• 使用 word2vec 工具箱在中文微博语料库中预先训练的词向量，总计 24,166 条记录，单词嵌入的维数设置为 200。

• 存在不命中的问题，即数据集中的词语没有出现在 w2v_200 的 24,166 条记录里，解决方法为随机初始化。

• 例如：训练使用的基于新浪新闻的 ECPE 语料库总计 24,165 个不重复的词语，命中的只有 12151 条：

```
load embedding...
w2v_file: ./data_combine/w2v_200.txt
all_words: 24165 hit_words: 12151
embedding.shape: (24166, 200) embedding_pos.shape: (201, 50)
load embedding done!
```

图 4.1：embedding 命中示意图

改进方法：

使用基于中文语料库预训练的 BERT 模型，动态生成训练、测试使用的数据集对应的所有词语的词向量。

BERT：Bidirectional Encoder Representation from Transformers[4]

BERT 是一种基于微调的多层双向 Transformer 编码器，主要利用了 Transformer 的 Encoder 结构，采用的是最原始的 Transformer。

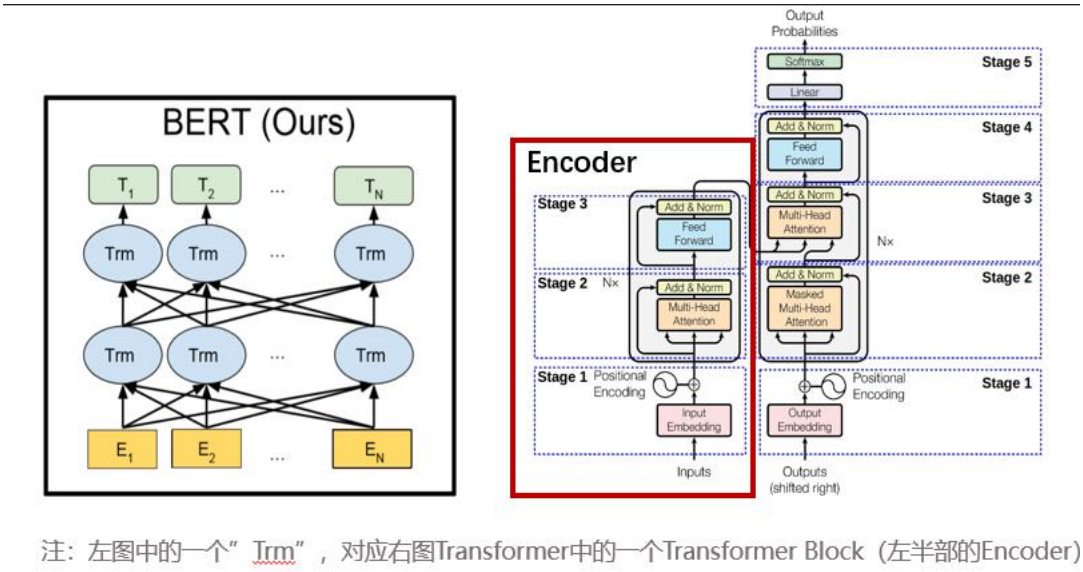

图 4.2：BERT 结构

4.2 BERT 的作用与优越性

• BERT 是一种预训练语言表示的方法，在大型文本语料库（例如 Wikipedia）上训练通用的“语言理解”模型，然后将该模型用于我们关心的下游 NLP 任务（例如问题回答、情感分类等）。

• BERT 本质上是一个两段式的 NLP 模型。第一个阶段叫做：Pre-training，与 Word Embedding 类似，利用现有无标记的语料训练一个语言模型。第二个阶段叫做：Fine-tuning，利用预训练好的语言模型，完成具体的 NLP 下游任务。

• BERT 优于以前的方法，因为它是第一个用于预训练 NLP 的无监督、深度双向系统。而无监督意味着 BERT 仅使用纯文本语料库进行培训，这很重要，因为在网络上可以多种语言公开获取大量纯文本数据。

4.3 BERT-wwm 预训练模型 & 全词 Mask 方法（wwm）

• 哈工大和科大讯飞联合发表于 2019 年的一篇论文，将全词 Mask 的方法应用在了中文之中。[5]

• 使用了中文维基百科（包括简体和繁体）进行训练，并且使用了哈工大 LTP 作为分词工具，即对组成同一个词的汉字全部进行 Mask。

• 将全词 MASK 训练方法运用于了中文的预训练模型之上，并没有对 BERT 模型结构进行修改。

全词 Mask 方法（wwm）

• 谷歌在 2019 年 5 月 31 日发布的一项 BERT 的升级版本，主要更改了原预训练阶段的训练样本生成策略。

• 在全词 Mask 中，如果一个完整的词的部分 Word Piece 子词被 mask，则同属该词的其他部分也会被 mask，即全词 Mask。

[Original Sentence] 使用语言模型来预测下一个词的probability。 **[Original Sentence with CWS]** 使用 语言 **模型** 来 **预测** 下 一个 词 的 **probability** 。
[Original BERT Input] 使 用 语 言 **[MASK]** 型 来 **[MASK]** 测 下 一 个 词 的 **pro [MASK] ##lity** 。 **[Whold Word Masking Input]** 使 用 语 言 **[MASK] [MASK]** 来 **[MASK] [MASK]** 下 一 个 词 的 **[MASK] [MASK] [MASK]** 。

表 4.1：全词 MASK 的生成样例

4.4 我们选用的 BERT 预训练模型

BERT-wwm & BERT-wwm-ext

• 都属于 BERT-base 模型：12-layer, 768-hidden, 12-heads, 110M parameters

• BERT-wwm, Chinese：基于中文维基百科（0.4B）

• BERT-wwm-ext, Chinese：基于 EXT 数据（中文维基百科，其他百科、新闻、问答等数据，总词数达 5.4B）

	BERT-wwm	BERT-wwm-ext
Masking	WWM[1]	WWM[1]
Type	base	base
Data Source	wiki	wiki+ext[2]
Training Tokens #	0.4B	5.4B
Device	TPU v3[3]	TPU v3
Training Steps	$100K^{MAX128}$	$1M^{MAX128}$
	$+100K^{MAX512}$	$+400K^{MAX512}$
Batch Size	2,560 / 384	2,560 / 384
Optimizer	LAMB	LAMB
Vocabulary	~BERT[4]	~BERT
Init Checkpoint	~BERT	~BERT

[1] WWM = Whole Word Masking

[2] ext = extended data

[3] TPU v3 (128G HBM)

[4] ~BERT 表示**继承**谷歌原版中文 BERT 的属性

图 4.3：BERT-wwm & BERT-wwm-ext

4.5 基于预训练 BERT 模型的词向量生成相关工作

第一步：

- 编写代码，**对基于 Google 开源的 BERT[6]代码进行了进一步的简化**，便于生成词向量/句向量
- 参考：bert-as-service ——腾讯 AI Lab 开源的 BERT 服务[7]

第二步：

- 每一层 transformer 的输出值，理论上来说都可以作为句向量，但根据前人的实验数据，**最佳结果是取倒数第二层**，最后一层的值太接近于目标，而前面几层语义还未被充分学习。

第三步：

- 编写程序，将需要生成词向量的数据集的词语汇总为 List，调用 **BERT-wwm 或 BERT-wwm-ext** 为其生成词向量，并将其用于之后的 ECPE 任务

（五）模型训练与参数设置

5.1 10 折交叉验证（10-fold Cross Validation）

防止过拟合，评估模型的泛化能力。

图 5.1：10 折交叉验证

使用这种方法，我们将数据集随机分成 10 份，使用其中 9 份用作训练而将另外 1 份用作测试，每次使用的测试数据不同。即该过程可以重复 10 次，训练完成这 10 个模型之后，然后针对每一个模型的测试集进行测试，最后得到可信的平均结果。

5.2 模型相关参数&训练设置

模型在 BiLSTM 中隐藏单元的数量设置为 100。所有的权重矩阵和偏差由 U(−0.01,0.01)均匀

分布随机初始化。
在训练细节上，我们使用随机梯度下降(SGD)算法和随机小批量的 Adam 更新规则。Batch Size 和 Learning Rate 分别设置为 32 和 0.005，Epoch 设置为 15
在正则化 Regularization 方面，word embeddings 应用 dropout, dropout 率设置为 0.8。此外，对 soft-max 参数进行 L2 约束，L2-norm 正则化(权重衰减)设置为 1e-5。

（六）实验评估及结果分析

6.1 实验评估标准

(1) 评估情绪提取和原因提取两个子任务的表现

以 Gui 等人(2016a)定义的精度、召回率和 F1 评分作为评价指标：

$$P = \frac{\sum correct\_positoins}{\sum proposed\_positons}$$

$$R = \frac{\sum correct\_positoins}{\sum annotated\_positons}$$

$$F1 = \frac{2 \times P \times R}{P + R}$$

$$P = \frac{\sum correct\_causes}{\sum proposed\_causes}$$

$$R = \frac{\sum correct\_causes}{\sum annotated\_causes}$$

$$F1 = \frac{2 \times P \times R}{P + R}$$

(2) 评估情感-原因子句对抽取总任务的表现

我们使用 precision、recall 和 F1 评分作为评估指标，计算公式如下：

$$P = \frac{\sum correct\_pairs}{\sum proposed\_pairs}$$

$$R = \frac{\sum correct\_pairs}{\sum annotated\_pairs}$$

$$F1 = \frac{2 \times P \times R}{P + R}$$

其中 proposed_pairs 表示模型预测的情感-原因对的数量，annotated_pairs 表示在数据集中标记的情感-原因对的总数，correct_pairs 表示既被标记又被预测的情感-原因对的数量。

6.2 实验结果及分析

Indep: 情感提取和原因提取分别由两个 Bi-LSTM 进行建模
Inter-CE: 原因提取的预测用于增强情感提取
Inter-EC: 情感提取的预测用于增强原因提取
-BERT: 使用 BERT-wwm 模型生成词向量
-BERT-ext:使用 BERT-wwm-ext 模型生成词向量

position_predict			
	P	R	F1
Indep	0.8476	0.8139	0.8299
Inter-CE	**0.8508**	**0.814**	**0.8317**
Inter-EC	0.8331	0.813	0.8227
Indep-BERT	0.861	0.8809	0.8696
Inter-CE-BERT	**0.8646**	0.8795	**0.8707**
Inter-EC-BERT	0.8501	**0.8885**	0.8677
Indep-BERT-ext	**0.8759**	0.858	0.8658
Inter-CE-BERT-ext	0.8636	0.8612	0.8613
Inter-EC-BERT-ext	0.8569	**0.8882**	**0.8718**

cause_predict			
	P	R	F1
Indep	0.6826	0.5711	0.6202
Inter-CE	0.6739	0.5626	0.6123
Inter-EC	**0.7113**	**0.6044**	**0.6523**
Indep-BERT	0.686	0.6053	0.6391
Inter-CE-BERT	0.6897	0.6085	0.6439
Inter-EC-BERT	**0.7275**	**0.6568**	**0.6891**
Indep-BERT-ext	0.7297	0.6108	0.6638
Inter-CE-BERT-ext	**0.7508**	0.6324	**0.6856**
Inter-EC-BERT-ext	0.7282	**0.636**	0.6748

图 6.1：情感预测&原因预测性能分析

- 与 Indep 相比，Inter-CE 在情感提取子任务上有了一些提升，改进主要体现在情感提取子任务的精度上，**证明了原因提取的监督有利于情感提取**。
- 与 Indep 相比，Inter-EC 在原因提取子任务上有了一些提升，改进主要体现在原因提取子任务的精度上，**证明了情感提取的监督有利于原因提取**。
- **采用 BERT 预训练模型生成词向量后，模型的精度和召回率均有较大的提升**，符合预期。总体上采用 BERT-wwm-ext 方法后提升效果更加显著。

emotion-cause_pair_predict							
	without emotion-cause pair filtering			with emotion-cause pair filtering			
	P	R	F1	P	R	F1	keep_rate
Indep	0.5765	0.5151	0.5428	0.6588	0.5127	0.5753	0.8716
Inter-CE	0.5887	0.5259	0.5535	0.6693	0.5226	0.5854	0.8742
Inter-EC	**0.5998**	**0.5602**	**0.5783**	**0.6761**	**0.5588**	**0.611**	0.8859
Indep-BERT	0.5461	0.5621	0.5481	0.7446	0.5522	0.6311	0.7199
Inter-CE-BERT	0.5583	0.5713	0.562	0.7309	0.5591	0.6316	0.7459
Inter-EC-BERT	**0.5811**	**0.6256**	**0.6006**	**0.7299**	**0.6175**	**0.6676**	0.7831
Indep-BERT-ext	0.6016	0.5551	0.575	0.7552	0.5464	0.6318	0.783
Inter-CE-BERT-ext	**0.6214**	0.5962	**0.6072**	**0.757**	0.5875	**0.6607**	0.8065
Inter-EC-BERT-ext	0.5891	**0.6069**	0.5933	0.7375	**0.5942**	0.6547	0.7807

图 6.2：情感原因配对性能分析

- 在情感-原因子句对抽取总任务上，**Inter-EC 比 Inter-CE 性能提升效果更好**。
- **采用 BERT 预训练模型生成词向量后，模型的精度和召回率均有较大提升**，符合预期。
- 有无过滤表示**在第二步应用笛卡尔积后是否采用过滤**。Keep-rate 表示经过过滤后最终保留的情感-原因对在 pair 中所占的比例。
- 使用过滤后所有模型在 ECPE 任务上 F1 得分得到了显著的改进，证明了**过滤器的有效性**。几乎所有模型的精度分数都有很大提高(超过 7%)，而召回率下降很小(小于 1%)，**导致 F1 的分数有显著提高**。

综上，结论为：

- 通过对 Inter-EC 和 Inter-CE 的比较，发现 **Inter-EC 的改善主要体现在原因提取任务**上，**Inter-CE 的改善主要体现在情感提取任务**上。这些结果与我们的直觉一致，即**情绪和原因是相互指示的**。此外，我们发现 **Inter-EC 在原因提取任务上的改善远远大于 Inter-CE 在情感提取任务上的改善**。我们猜测这是因为提取原因比提取情感更困难，所以有更多的改进空间。
- **采用 BERT 预训练模型生成词向量后，所有方法均相较于采用静态 word2vec 均有较大的提升**，符合我们的预期；**BERT-wwm-ext 的效果更好，**因为该模型的预训练基于 EXT 数据（中文维基百科，其他百科、新闻、问答等数据，总词数达 5.4B），**训练语料更丰富**。
- 使用过滤后所有模型在 ECPE 任务上 F1 得分得到了显著的改进，证明了**过滤器的有效性**

（七）ECPE-Demo 使用说明

7.1 所需环境

python3.7,　tensorflow1.14.0, numpy1.16.0

7.2 使用方法

(1) 本模型的输入文件为 input/data.txt，将待预测的文本复制到该文件中，其中待预测的文件已经放在了 data 目录下(论文的训练集和测试集)。图 7.1 展示了输入文件中某段文本。第一行(423, 10)分别代表这段文本的 id，共 10 个句子。第二行的(5, 4)代表第五句是情感句，

```
423 10
 (5,4)
1,null,null,我们 结婚 吧
2,null,null,这样 别人 再 问
3,null,null,你 就 有 身份 了
4,null,null,看到 桓江为 自己 和 家人 奔走 忙碌
5,happiness,感动,李春霞 十分 感动
6,null,null,桓江 当即 表态
7,null,null,会 和 李春霞 一起 照顾 米全
8,null,null,绝不 离弃
9,null,null,2011 年 5 月 31 日
10,null,null,李春霞 与 桓江 正式 登记 结婚
```

对应的原因句是第四句。

图 7.1：输入文件中某段文本

```
423 10
[(5, 4)]
1, 0, 0, 我们 结婚 吧
2, 0, 0, 这样 别人 再 问
3, 0, 0, 你 就 有 身份 了
4, 0, 1, 看到 桓江为 自己 和 家人 奔走 忙碌
5, 1, 0, 李春霞 十分 感动
6, 0, 0, 桓江 当即 表态
7, 0, 0, 会 和 李春霞 一起 照顾 米全
8, 0, 0, 绝不 离弃
9, 0, 0, 2011 年 5 月 31 日
10, 0, 0, 李春霞 与 桓江 正式 登记 结婚
```

图 7.2：情感句原因句预测结果

(2) 运行 Code 目录下的 emo_cause_predict.py 程序对输入文件中的情感句和原因句的位置进行预测，预测结果在 output/emo_cause/predict.txt 中查看。图 7.2 展示了输出文件中该段对应文本的预测结果。其中(423, 10)表示该段文本的 id，共有 10 个句子。[(5, 4)]是正确的情感原因句位置配对结果。第五句的第二列为 1 表示第五句是情感句，第四句的第三列为 1 表示第四句为原因句。

(3) 运行 Code 目录下的 pair.py 程序对预测文件中的情感句和原因句配对。图 7.3 展示了文件部分配对结果。每一行的第一个数字是段落的 id 号。“||”左边是正确的配对结果，右边是模型配对结果。前面提到的 id 为 423 的文本，情感原因句匹配正确。当然，一段文本中可能有不同情感，对应不同的原因，或者一个情感多种原因，如 id 为 429 的文本匹配结果所示。第 14 句是情感句，对应的原因句有两句，第 12 句和第 13 句。

```
423 (5, 4) || (5, 4)
424 (8, 7) || (8, 7)
425 (12, 12) || (12, 12)
426 (12, 11) || (12, 11)
427 (13, 12) || (13, 12)
428 (4, 3) || (4, 3)
429 (14, 12) (14, 13) || (14, 12) (14, 13)
430 (8, 1) || (2, 1)
```

图 7.3：配对结果

系统应用及可视化分析

8.1 各种情感原因词云

得到了标注的情感句和原因句后，利用大连理工大学的情感词汇本体库[8]，该库包含了27467 个中文情感词或情感短语，并对每个情感词或短语进行了情感分类，将情感分为了 7 个大类 21 个小类，并对情感强度、极性和褒贬等进行了标注，图 8.1 便是类别图，我们在实验中只用到了它的 7 个大类（乐、好、怒、哀、惧、恶、惊），我们将识别的情感句作为输入，通过与库中情感词对比来对情感句中情感词进行分类，进一步的，通过预测到的情感句与原因句的对应关系，可以对不同类别的原因句进行分类，图二是提取出的情感词和它所对应的原因句，在图 2 的基础上就能得到不同类别情感的原因的集合了。

编号	情感大类	情感类	例词
1	乐	快乐(PA)	喜悦、欢喜、笑眯眯、欢天喜地
2		安心(PE)	踏实、宽心、定心丸、问心无愧
3	好	尊敬(PD)	恭敬、敬爱、毕恭毕敬、肃然起敬
4		赞扬(PH)	英俊、优秀、通情达理、实事求是
5		相信(PG)	信任、信赖、可靠、毋庸置疑
6		喜爱(PB)	倾慕、宝贝、一见钟情、爱不释手
7		祝愿(PK)	渴望、保佑、福寿绵长、万寿无疆
8	怒	愤怒(NA)	气愤、恼火、大发雷霆、七窍生烟
9	哀	悲伤(NB)	忧伤、悲苦、心如刀割、悲痛欲绝
10		失望(NJ)	憾事、绝望、灰心丧气、心灰意冷
11		疚(NH)	内疚、忏悔、过意不去、问心有愧
12		思(PF)	思念、相思、牵肠挂肚、朝思暮想
13	惧	慌(NI)	慌张、心慌、不知所措、手忙脚乱
14		恐惧(NC)	胆怯、害怕、担惊受怕、胆颤心惊
15		羞(NG)	害羞、害臊、面红耳赤、无地自容
16	恶	烦闷(NE)	憋闷、烦躁、心烦意乱、自寻烦恼
17		憎恶(ND)	反感、可耻、恨之入骨、深恶痛绝
18		贬责(NN)	呆板、虚荣、杂乱无章、心狠手辣
19		妒忌(NK)	眼红、吃醋、醋坛子、嫉贤妒能
20		怀疑(NL)	多心、生疑、将信将疑、疑神疑鬼
21	惊	惊奇(PC)	奇怪、奇迹、大吃一惊、瞠目结舌

图 8.1：情感分类库展示

在得到了情感原因的分类后，就可以统计不同情感的原因中各个词出现的频率，进一步的，就可以根据频率得到词云和气泡图，值得一提的是，因为原因文本中，有很多无意义的词，比如“的”“他们”等，我们利用哈尔滨工业大学整理的中文停用词的库对原因句中的词进行筛选，这样得到的词的频次才有意义。图 8.2 给出了 7 大类中哀伤和快乐两类的词云图，可以看到，除了一些情感词外，最显眼的就是“儿子”“女儿”“丈夫”等关系词，现实生活中也是他们和我们的生活关系最密切，所以情感原因与他们的关联度比较大。图 8.3 给出了愤怒和恐惧的气泡图。

图 8.2：哀伤(左)和快乐(右)的词云

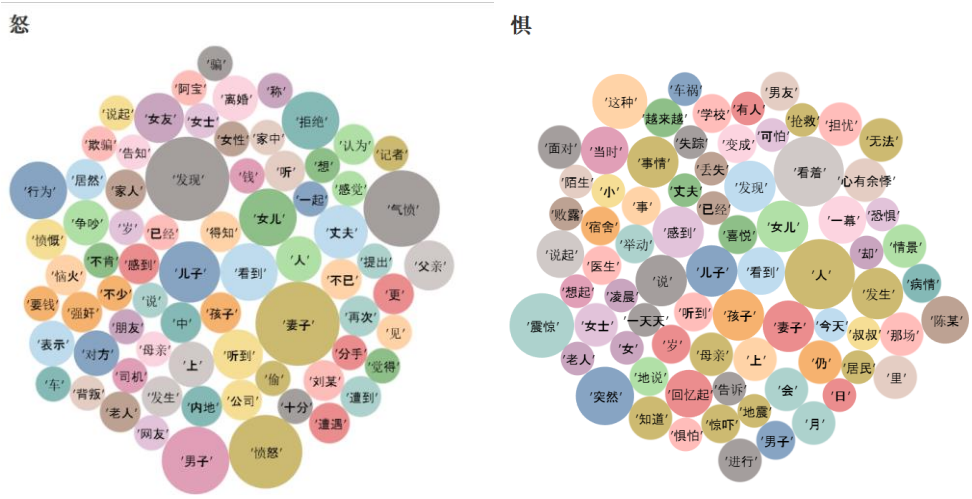

图 8.3：愤怒(左)和恐惧(右)的气泡图

8.2 抑郁原因分析

将训练好的模型用于从抑郁吧爬取的数据，尝试从爬取的文本中得到抑郁的原因，图 8.4 和图 8.5 都是预测结果的部分截图，在第 187 段文本中有 7 个分句，其中第 3 句的原因句标志位为 1，表示这一句是抑郁原因句，同样，图 8.5 的第 6 句也是抑郁原因句。希望能从大量抑郁文本中统计抑郁原因，对抑郁症的研究起到帮助作用，但由于爬取文本的格式与原有数据集差距较大，所以在爬取的抑郁数据上的预测结果有待进一步改善。若使用内容丰富的抑郁原因标注数据集作为训练数据，相信该系统的性能会得到进一步的提升。

```
189 7
1, 0, 0, 一句 话
2, 0, 0, 又 把 我 拉回来 了
3, 0, 1, 搞 啥 也 不 知道 自问自答
4, 0, 0, 自言自语
5, 0, 0, 疯 的 安静 够 聪明 不会 做 傻事
6, 0, 0, 傻得 彻底
7, 0, 0, 不 知道 自己 做 傻事
```

图 8.4：抑郁文本中原因句提取(例 1)

```
468 9
1, 0, 0, 不妨 改变
2, 0, 0, 改变 状态
3, 0, 0, 改变 思维 角度
4, 0, 0, 千万 不要 一条 路 走 到 黑 的话
5, 0, 0, 反而 看不见 黎明
6, 0, 1, 人心 有时候 很难 自我调节 和 控制
7, 0, 0, 需要 外力 感染 和 干扰 环境 他人
8, 0, 0, 心理咨询 就是 帮助 走出 困境 的 人
9, 0, 0, 人 是 最 厉害 的 什么样 的 生活 都 总有一天 会 坦然 接受 一个 人
```

图 8.5：抑郁文本中原因句提取(例 2)

引用参考

[1] Gui, Lin, et al. "A question answering approach to emotion cause extraction." *arXiv preprint arXiv:1708.05482* (2017).

[2] Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu, and Yu Zhou. 2016a. Event-driven emotion cause extraction with corpus construction. In Empirical Methods in Natural Language Processing (EMNLP), pages 1639–1649.

[3] Xia R , Ding Z . Emotion-Cause Pair Extraction: A New Task to Emotion Analysis in Texts[J]. 2019.

[4] Devlin J , Chang M W , Lee K , et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. 2018.

[5] Cui, Yiming, et al. "Pre-training with whole word masking for chinese bert." *arXiv preprint arXiv:1906.08101* (2019). https://arxiv.org/abs/1906.08101

[6] Turc, Iulia, et al. "Well-read students learn better: On the importance of pre-training compact models." arXiv preprint arXiv:1908.08962 (2019).

[7] 腾讯 AI Lab 开源的 BERT 服务 https://github.com/hanxiao/bert-as-service

[8] 大连理工大学情感词汇本体库 https://github.com/ZaneMuir/DLUT-Emotionontology