AIChallenger2018冠军代码PPT分享--细粒度情感分析赛道

专知 (/users/10000031042) 2018-12-26 15:06

关注文章

分享心得 (/articles?tagId=166945)

转载自：**AINLP**

赛题简介：在线评论的细粒度情感分析对于深刻理解商家和用户、挖掘用户情感等方面有至关重要的价值，并且在互联网行业有极其广泛的应用，主要用于个性化推荐、智能搜索、产品反馈、业务安全等。本次比赛我们提供了一个高质量的海量数据集，共包含6大类20个细粒度要素的情感倾向。参赛人员需根据标注的细粒度要素的情感倾向建立算法，对用户评论进行情感挖掘，组委将通过计算参赛者提交预测值和场景真实值之间的误差确定预测正确率，评估所提交的预测算法。

冠军介绍:程惠阁，硕士毕业于北京大学计算机系人机交互与多媒体实验室，曾负责百度贴吧/feed反作弊及图片搜索部图文方向，2017AI Challenge图像中文描述赛道亚军、

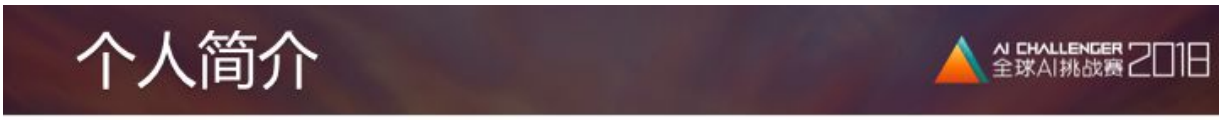
Github地址：

https://github.com/chenghuige/wenzheng/tree/master/projects/ai2018/sentiment (https://github.com/chenghuige/wenzheng/tree/master/projects/ai2018/sentiment)

代码使用几点注意事项：

- 主要参考algos（tf模型）实现版本和torch-algos（pytorch实现版本）
- python path需要设置 下载路径utils 这样能找到下面的melt等路径

PPT内容如下：文末打包下载



- 硕士毕业于北京大学计算机系人机交互与多媒体实验室
- 曾负责百度贴吧/feed反作弊及图片搜索部图文方向
- 2017 ai challenger 图像中文描述赛道亚军
- 2018 kaggle toxic comments classification challenge 第12名/4551

Testb结果

#	团队名称	成员	最好成绩	提交次数	上次提交
1	后厂村静静		0.72946	2	2018/11/15 21:21:30
2	do something		0.72794	2	2018/11/15 21:35:36
3	simtony		0.72736	2	2018/11/15 20:43:58
4	原来在这里改名字的		0.72681	2	2018/11/15 20:05:50
5	Artificial_Idiot		0.72677	2	2018/11/15 19:54:55

大纲

- 依赖与参考
- 数据
- 技术细节
- 基线模型
- 基于MRC的文本分类模型
- 基于bert的文本分类模型
- 模型集成
- 整体迭代过程 & TODO

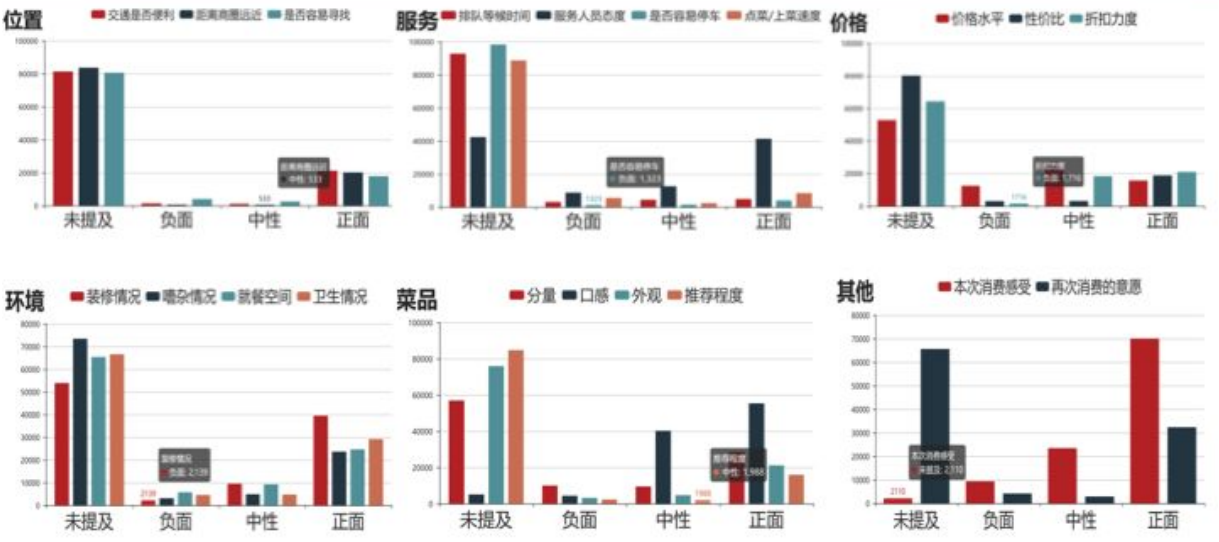
依赖和参考

- 主要依赖：
  - Tensorflow 1.10.1(数据预处理和模型)
  - Pytorch 1.0(模型)
- 主要代码参考：
  - https://github.com/HKUST-KnowComp/R-Net
  - https://github.com/HKUST-KnowComp/MnemonicReader
  - https://github.com/google-research/bert
  - https://github.com/allenai/allennlp/

HKUST-KnowComp
Knowledge Computation Group@HKUST, led by Yangqiu Song

- 外部数据(只用来预训练词向量和语言模型)：
  - https://github.com/SophonPlus/ChineseNlpCorpus

数据分布



数据预处理

- 繁体转简体
  - 『精伍鴨頭湖北菜館』位於環球大廈一樓西側1054號，服務電: 88259300，位於鬧市區，不太好停車
  - 『精伍鸭头湖北菜馆』位於环球大厦一楼西侧1054号，服务电: 88259300，位於闹市区，不太好停车
  - 影响面 4.61% (4839/105000)
- 特殊表情字符不被切分
  - Emoji (pyemoji) 😄
  - 高频表情

    点评 ^_^ 。

数据预处理

- 五种分词方式
  - 结巴分词

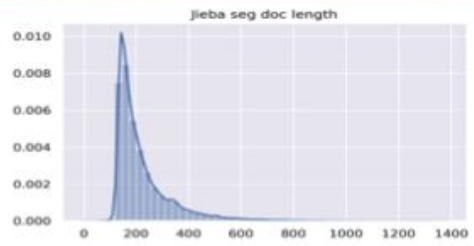
  - Unigram Language Model
    - 1w

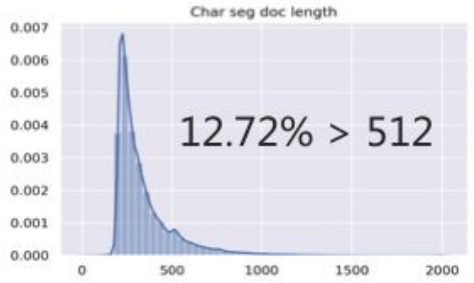
    - 10w

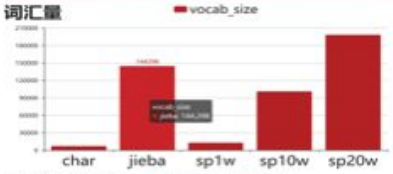
    - 20w

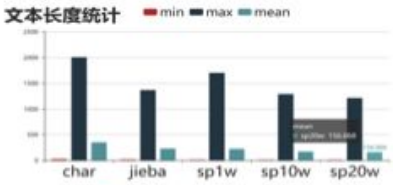
  - 基于char的切分

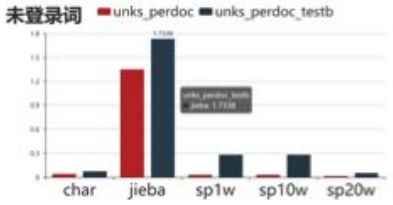
不同分词方式对比


Jieba seg doc length
0.010
0.008
0.006
0.004
0.002
0.000
0
200
400
600
800
1000
1200
1400


Char seg doc length
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
500
1000
1500
2000
12.72% > 512


词汇量
vocab_size
char
jieba
sp1w
sp10w
sp20w


文本长度统计
min
max
mean
char
jieba
sp1w
sp10w
sp20w


未登录词
unks_perdoc
unks_perdoc_testb
char
jieba
sp1w
sp10w
sp20w

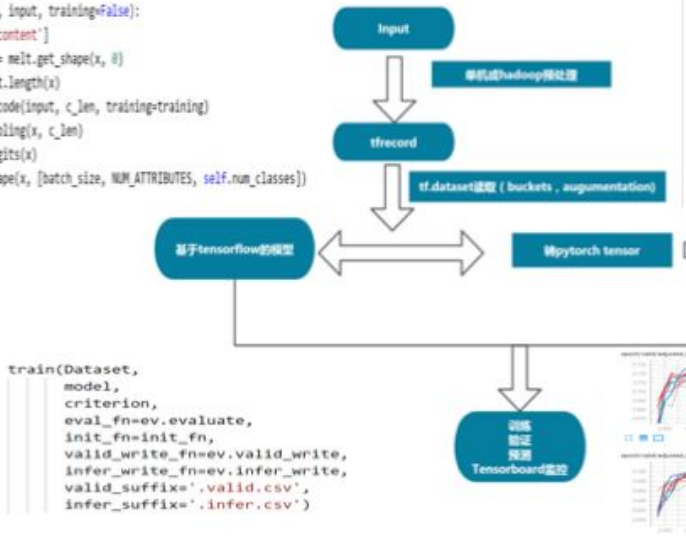
数据


comment


点评评论数据
1kw
无监督数据
https://github.com/SophonPlus/ChineseNlpCorpus
标注数据
10.5w
官方训练数据
官方验证数据数据
1.5w

实验框架


```
class Model(ModelBase):
  def call(self, input, training=False):
    x = input['content']
    batch_size = melt.get_shape(x, 0)
    c_len = melt.length(x)
    x = self.encode(input, c_len, training=training)
    x = self.pooling(x, c_len)
    x = self.logits(x)
    x = tf.reshape(x, [batch_size, NUM_ATTRIBUTES, self.num_classes])
    return x
```




```
class Model(ModelBase):
  def forward(self, input, training=False):
    x = input['content']
    x_mask = x.eq(0)
    batch_size = x.size(0)
    x = self.encode(input, x_mask, training=training)
    x = self.pooling(x, x_mask)
    x = self.logits(x)
    x = x.view([-1, NUM_ATTRIBUTES, self.num_classes])
    return x
```

```
train(Dataset,
      model,
      criterion,
      eval_fn=ev.evaluate,
      init_fn=init_fn,
      valid_write_fn=ev.valid_write,
      infer_write_fn=ev.infer_write,
      valid_suffix='.valid.csv',
      infer_suffix='.infer.csv')
```

F1指标的优化

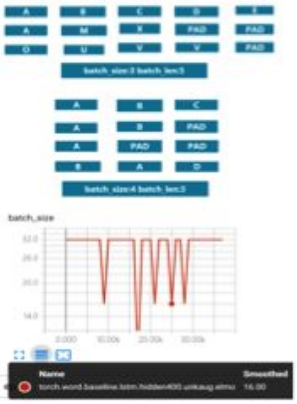
- 不调整训练loss
- 不对训练数据做过/欠采样
- 只针对训练结果按照类别在train数据的分布做调整
  - $logits = logits * 10$
  - $probs = softmax(logits)$
  - $adjusted\_probs = probs * (total/freq(clas$
  - $predicts = argmax(adjusted\_probs)$

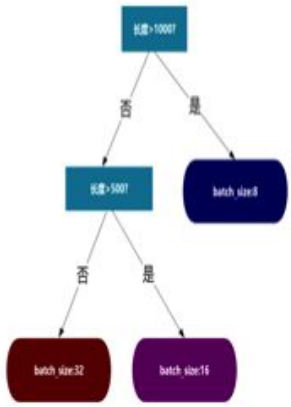
F1 ↑0.6-0.8%

Valid	调整后F1	原始F1
单模型	0.71930	0.71243
集成	0.72874	0.72092

技术细节-batch size

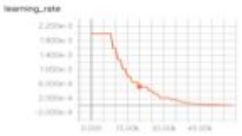
- 长度分桶(bukcets length)
  - 避免长文本截断同时不影响整体训练速度
  - 按文本长度聚簇
    - 长文本的小的batch size
    - 短文本采用大batch size
    - buckets=500,1000
    - batch_sizes=32,16,8





技术细节-学习率

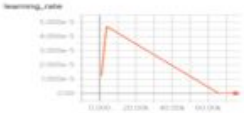
- 学习率调节方法
  - 基于valid数据的自适应学习率下降



三角学习率

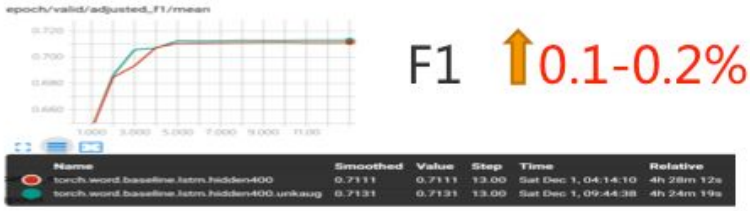
Loss ↓1%

- 三角学习率
  - Warm up
  - Linear lr decay



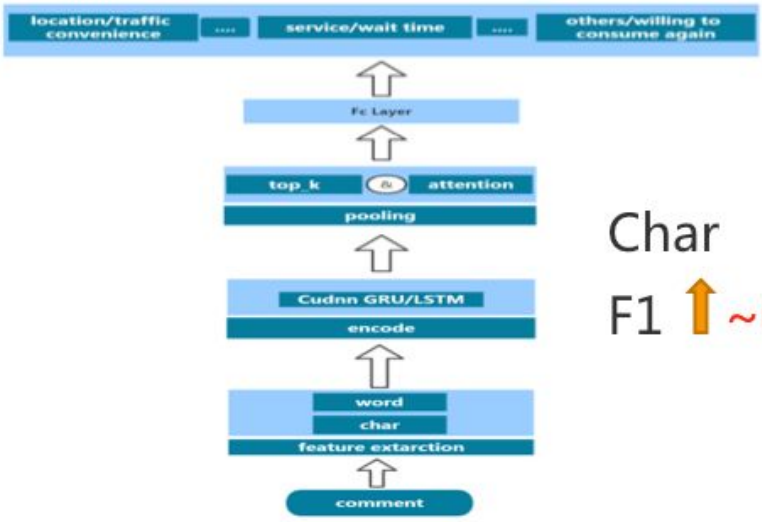
技术细节-数据增强

- 随机设置UNK
  - 从第二轮开始，以0-0.02的随机概率设置原词为UNK



F1 ⬆0.1-0.2%

基线模型-End2End分类模型



Char

F1 ⬆~0.5%

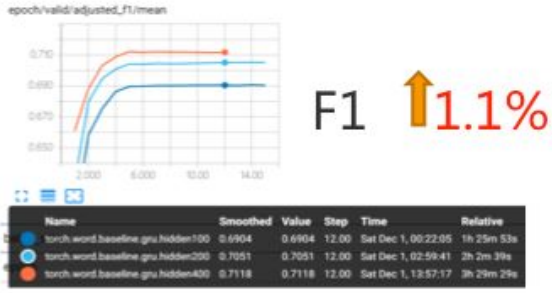
基线模型-参数

Baseline	Tensorflow	Pytorch
Optimizer	Adam	Adamax
Num layers	2	2
Hidden size	400	400
Dropout rate	0.3	0.3
Concat layers	Yes	No
Recurrent dropout	Yes	No
Rnn padding	Yes	No
Pooling	Top3 + Linear Attention	Top3 + FFN attention

基线模型-参数

- 参数多的模型效果更好(The wider the better)
  - Hidden size
    - 400 > 200 > 100

F1 ↑0.6%



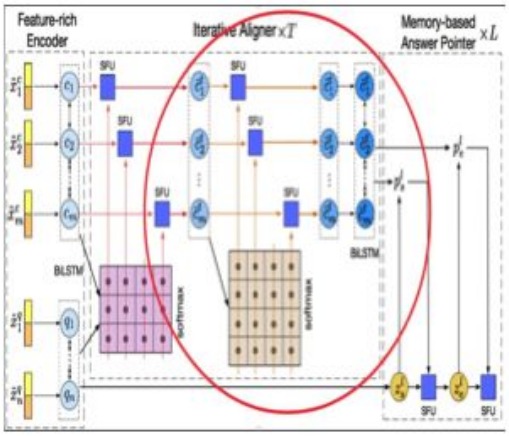
F1 ↑1.1%

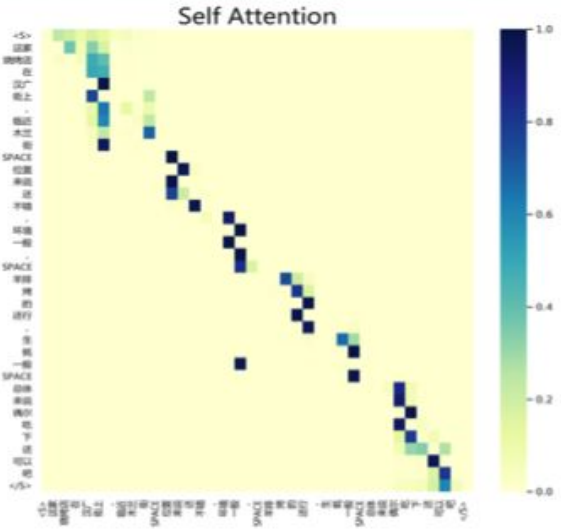
MRC模型- Self Attention


```
MReader(
  (encode): TextEncoder(
    (embedding): Embedding(144299, 300, padding_idx=0)
    (char_embedding): Embedding(6798, 300, padding_idx=0)
    (char_encode): StackedBRNN(
      (rnns): ModuleList(
        (0): LSTM(300, 400, bidirectional=True)
      )
    )
    (char_pooling): Pooling(
      (poolings): ModuleList(
        (0): LastPooling()
      )
    )
    (encode): StackedBRNN(
      (rnns): ModuleList(
        (0): LSTM(1100, 400, bidirectional=True)
        (1): LSTM(800, 400, bidirectional=True)
      )
    )
  )
  (pooling): Pooling(
    (poolings): ModuleList(
      (0): TopKPooling(
      (1): NonLinearSeqAttnPooling(
        (FFN): FeedForwardNetwork(
          (linear1): Linear(in_features=800, out_features=128, bias=True)
          (linear2): Linear(in_features=128, out_features=1, bias=True)
        )
      )
    )
  )
  (logits): Linear(in_features=3200, out_features=80, bias=True)
  (interactive_aligners): ModuleList()
  (interactive_SFUs): ModuleList()
  (self_aligners): ModuleList(
    (0): SelfAttnMatch()
  )
  (self_SFUs): ModuleList(
    (0): SFU(
      (linear_r): Linear(in_features=3200, out_features=800, bias=True)
      (linear_g): Linear(in_features=3200, out_features=800, bias=True)
    )
  )
  (aggregate_rnns): ModuleList(
    (0): StackedBRNN(
      (rnns): ModuleList(
        (0): LSTM(800, 400, bidirectional=True)
      )
    )
  )
)
```

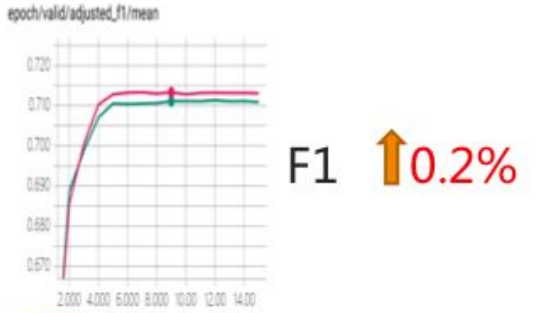
MRC	Tensorflow(Rnet)	Pytorch(Mreader)
Attention combiner	Gate	Semantic Fusion(SFU)

MRC模型 – Self Attention

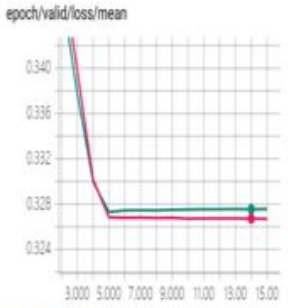




MRC模型– Self Attention效果



F1 ↑0.2%

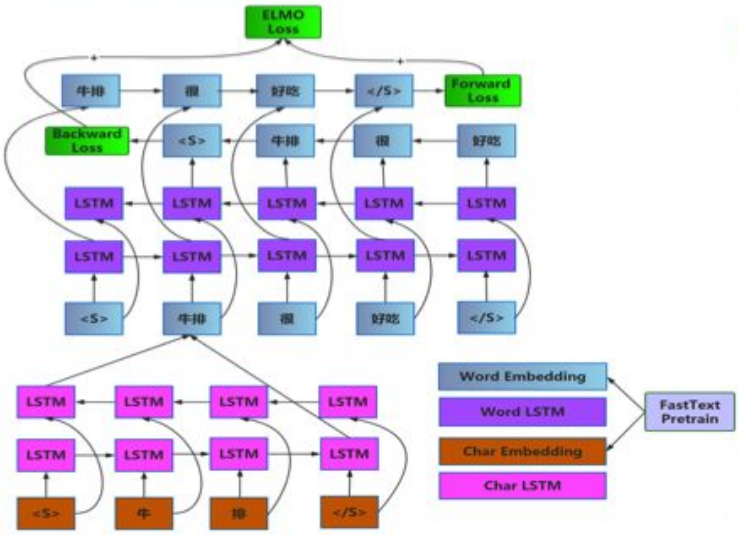


Loss ↓0.1%

Name	Smoothed	Value	Step	Time	Relative
torch.self_attention.unkaug	0.7133	0.7133	9.000	Mon Dec 3, 02:35:00	5h 54m 41s
torch.word.baseline.lstm.hidden400.unkaug	0.7111	0.7111	9.000	Sun Dec 2, 23:53:26	3h 29m 28s

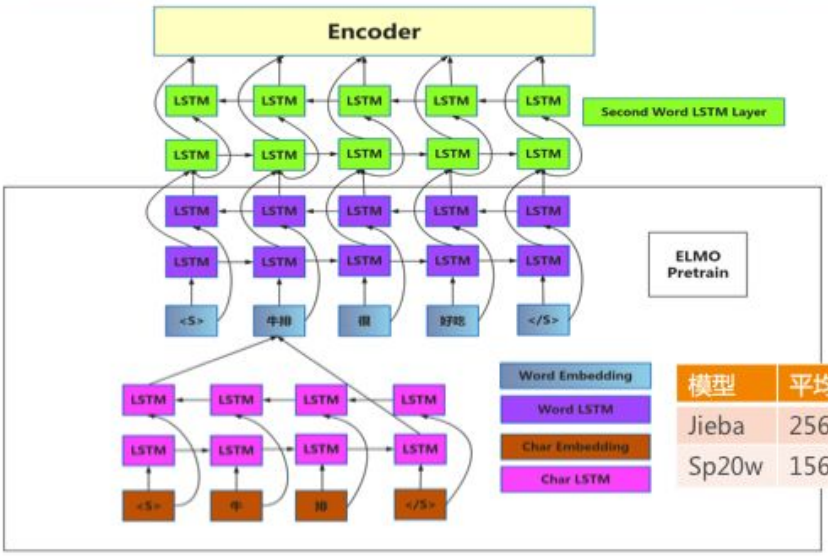
Name	Smoothed	Value	Step	Time	Relative
torch.self_attention.unkaug	0.3267	0.3267	14.00	Mon Dec 3, 06:16:34	9h 36m 15s
torch.word.baseline.lstm.hidden400.unkaug	0.3276	0.3276	14.00	Mon Dec 3, 02:04:08	5h 40m 9s

Fast ELMO-预训练



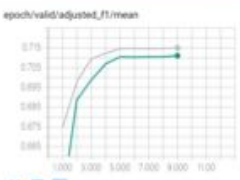
	说明
评论数目	1kw
FastText预训练(5轮)	5hour
Batch size	32
评论长度	256
Sample Softmax	No
Hidden size	400
词表大小	14.4w
ELMO 预训练(1轮)	45hour
ELMO 需要轮次	0.1-1

Fast ELMO-Finetune



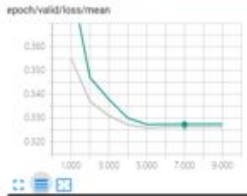
模型	平均长度	1轮时间	最佳模型
Jieba	256	0.66hour	3.5hour
Sp20w	156	0.54hour	3hour

Fast Elmo – 效果

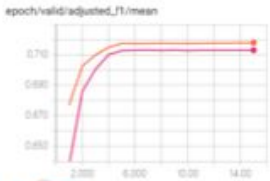


基线模型

F1 ↑ 0.4%

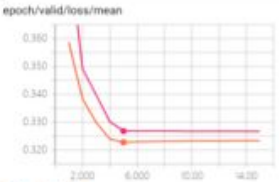


Loss ↓ 0.1%



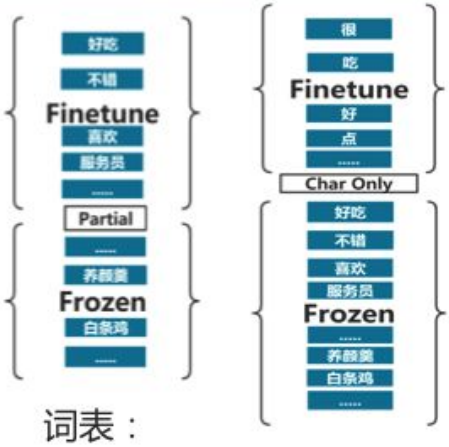
MRC模型

F1 ↑ 0.5%



Loss ↓ 0.4%

词嵌入- 部分Finetune



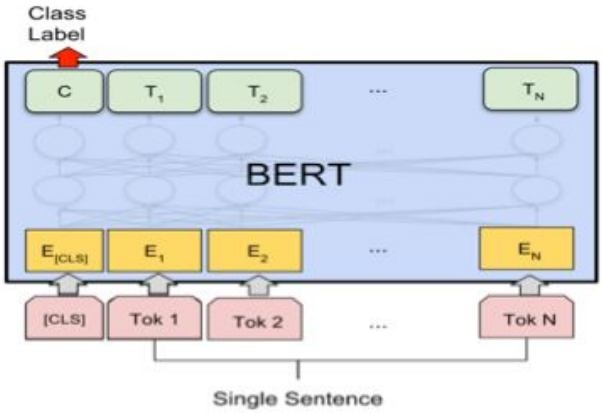
词表：

在所有语料(训练+无监督)

频次 >= 20

Finetune?(jieba)	Adjusted F1	Loss
No	0.7183	0.3228
Char Only	0.7198	0.3226
Partital(6k)	0.7197	0.3220
All(14.4w)	0.7204	0.3225
Finetune? (sp20w)	**Adjusted F1**	**Loss**
No	0.7155	0.3240
Char Only	0.7188	0.3235
Partitial(6k)	0.7180	0.3248
All(19.8w)	0.7172	0.3258

Bert模型



(b) Single Sentence Classification Tasks: SST-2, CoLA

Bert模型

- 文本截断
  - 保留尾部信息

857,"""之前看评论说味道还不错，所以找了一个时间和朋友去试一儿，觉得味道很好，所以想试一下这家的味道怎么样，结果非常让少十块钱，他还向我解释是他听签签数量听错了，但是如果你不让然的话就会多收你的钱。反正这家[illegible]感觉很一般，以后不会再去了

- 在点评语料finetune语言模型

valid	调整后F1	F1	Loss
Not finetune	0.70309	0.69759	0.35784
Finetune	0.70867	0.70178	0.35344
Delta	+0.55%	+0.25%	-0.44%

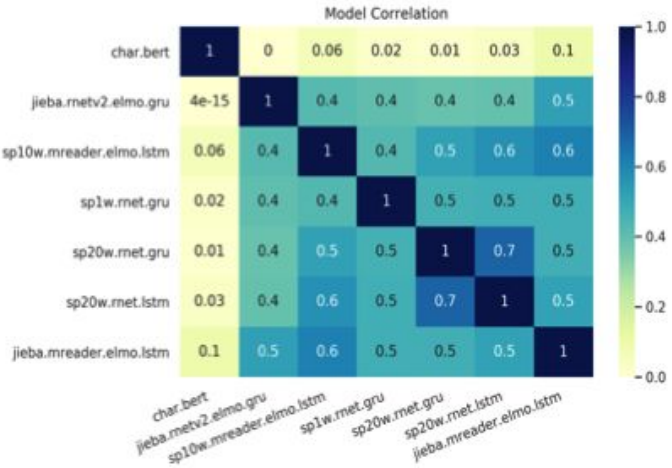
集成-7个单模型

模型(Valid)	调整后F1	原始F1	Loss
Jieba.mreader.elmo.lstm	0.71930	0.71243	0.32280
Sp20w.rnet.lstm	0.71798	0.71229	0.32748
Sp10w.mreader.elmo.lstm	0.71778	0.71207	0.32513
Jieba.rnetv2.elmo.gru	0.71720	0.71400	0.33610
Sp20w.rnet.gru	0.71714	0.71190	0.3317
Sp1w.rnet.gru	0.71654	0.70980	0.33390
Char.bert	0.70867	0.70178	0.35344

集成-模型差异性

- 模型
  - Rnn
    - Tensorflow版本Rnet
    - Pytorch版本Mreader
  - Transformer(bert)
- 分词方式
  - Char, Jieba, Sp1w,Sp10w,Sp20w
- 预训练
  - 是否使用elmo
- 数据
  - Bert Google的大规模预训练语料

集成-模型相关性



	Jieba.mreader.elmo.lstm
Sp10w.mreader.elmo.lstm	0.98836
Jieba.rnetv2.elmo.gru	0.98558
Sp20w.rnet.lstm	0.98538
Sp1w.rnet.gru	0.98403
Sp20w.rnet.gru	0.98399
Char.bert	0.97275

集成-效果(分类别Blending)

AI CHALLENGER 全球AI挑战赛 2018

F1 ↑ 0.8%

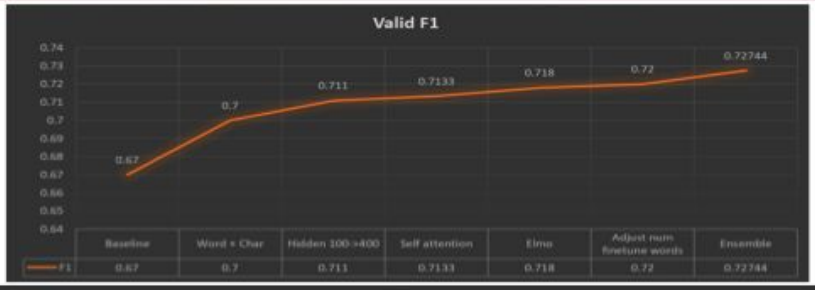
Loss ↓ 0.92%

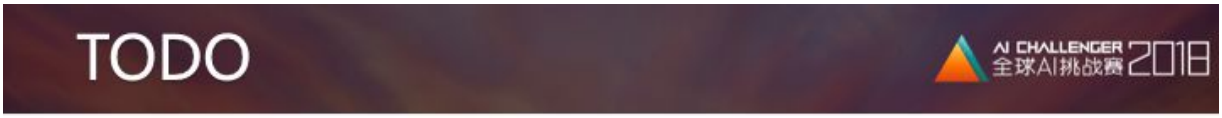
加入BERT Loss ↓ 0.1%

集成	Valid调整后F1	Valid F1	Valid Loss	TestB F1
7模型集成（10交叉)	0.72744	0.72004	0.31364	NA
7模型集成	0.72874	0.72092	0.31346	**0.729522**
去掉BERT（10交叉)	0.72750	0.71999	0.31459	NA
去掉BERT	0.72844	0.72068	0.31442	NA

整体迭代过程

AI CHALLENGER 全球AI挑战赛 2018





- 更好的F1优化策略，如强化对抗学习
- 优化bert的效果
  - 优化当前基于字的模型(512限制,batch size调大等等)
  - 基于词的bert模型
  - 基于词(transformer)+字(rnn/cnn)的bert模型
- 优化elmo的效果
  - elmo loss增加或改用bert loss
  - 支持Sampled softmax
- 尝试更好的集成策略

模型

暂无文件

推荐阅读

实习商汤，校招华为，我的深度学习之路 (/articles/3853?from=articles_commend)

评论

按时间倒序

支持Markdown和数学公式,公式格式：\\(...\\)或\\[...\\]

预览

你有什么看法?

还没有内容

联系我们

欢迎来到TinyMind。

关于TinyMind的内容或商务合作、网站建议，举报不良信息等均可联系我们。

TinyMind客服邮箱：support@tinymind.com.cn

TinyMind客服微信：tinymind01　　工作时间：周一至周五10:00-18:30