提交注意：保存成后缀txt文件，其实是tsv。Header手动删除。

1：BIDAF0 p=100 batch=256 gmp test1=0.6864

2：BIDAF0 p=150 batch=256 test2=0.6929 说明p取150更好。

3：BIDAF0 p=100 batch=128 spatial dropout=0.2

only1 modelling layer, 1 dense layer

val 0.8026 test3=0.7023 val和test基本成正比

4：在3的基础上batch=64 val=0.8017 批次小反而变差，用大batch。

改动数据预处理，分词词典根据本条的alternative即时调整，更加准确。

5：在3的基础上 batch=256 p=150

val=0.8038

6：在5的基础上数据增强。0.1的概率随机删除训练集passage里的词。Val=0.8011放弃增强

7：BIDAF1：改进AF层。自定义AF层加入α和β权重（变成additive attention），对最后的query aware representation进行self-attention。1 modelling layer。Gmp+gap+last hidden。1 dense layer。

时间太长，且耗内存Batch=128 p=100 要40分钟一代 。val=0.8038。模型太复杂而且无进步。

8：BIDAF2：在5的基础上，增加最后的query aware representation的self-attention。仍然采用简单的multiply attention即无权重。1 modelling layer。增加Gmp+gap+last hidden。1 dense layer。batch=256 p=150 rnn=64。 val=0.8033

9：BIMPM val=0.7950

10：在8的基础上rnn从64改到128。 Val=0.8064 说明rnn取128最好。

Rnn=256， val=0.8013 更复杂不好。

11：在8 的基础上Pad或者truncate改到句子末端。Val=0.8014。

在10 的基础上只将query的Pad或者truncate改到句子末端。Val=0.8060 差不多。为了防止q的区分词被截断，下面再次预处理以后就改为p：前端，q：后端

12：在8的基础上q不进行spatial Dropout并且把p的drop prob降到0.1。val=0.8001

在8的基础上q不进行spatial Dropout并且把p的drop prob提高到0.3。val=0.7974

说明p，q都drop 0.2最好

13：在8的基础上把self-attention移到p。val=0.8030。不好。

追加passage数据预处理。参考SAN模型增加feature（在词向量基础上增加exact match（1d），pos-tag（7d）， soft-align）。保存成 [sample, 150, 8]的ndarray。在预处理阶段就要对passage进行前端pad或truncate，这样不方便再调整passage长度。

14：BIDAF3：在增加feature的数据上进行8。Val=0.7993。 居然变差！！！

保留exact match和pos tag。Val=0.8077。说明soft align很差。

只保留exact match。Val=0.8098。

Exact match和option match。Val=0.8099。说明exact match很有用，option match作用很小。

Exact match和option match和cos sim。Val=0.8107。有用，保留。

由于15的发现，更改预处理长度后重新测试exact match。 p=200，q=15，Val=0.8083变差。

对q也加上exact match，option match和cos sim，q=15，val=0.8214.一个百分点的提高！！！

改用HanLP分词，歧义也能分开。但是单字更倾向成词，降低同义词匹配度？同一个词在p和

q分的不一致。val=0.8176，比上面的差。

细分问题形式，query和alternatives的merge结果更准确。Val=0.8210，没提高，但是好像过拟合减轻了。

先merge再分词，merge结果更自然。Val=08221。更好。详见preprocess2。Test=0.7283。

15：在8的基础上增加p到200，q到15。val= 0.8050。更好。但是不采纳，可能是稀释了exact match的效果？

16：BIDAF4：在14只保留exact match的基础上增加fasttext词向量。Val= 0.8122融合词向量好！

17：在16的基础上rnn=128，增加option match。训练到第3代loss最小0.4161，acc=0.8075，test16=0.7074。训练到第5代acc最高0.8125，test17=0.7184。如果训练到第6代Val=0.8101已经开始回落，test15=0.7122。说明acc回落确实已经过拟合。最好的模型应该是val acc最高的一次，保存其权重。

18：HAFN0：和BIDAF3保留exact match参数相同。Val=0.8050<0.8098 略差于BIDAF

19：HAFN0：和17参数相同。第4代loss最小loss=0.4141，Val=0.8094。最后一代loss=0.4446，val=0.8102，test19=0.7121

平均15和19，test=0.7273。融合有用。

20：r-net0：简化的HAFN。保留exact match和option match参数相同。Val=0.8039<0.8099 略差于BIDAF和HAFN。

研究valid 结果，前50个里犯错的，有1/3的错误是无法确定没有分出。没有误报无法确定的。

有一个q截取的太短。有一个分词严重错误。

未来改进点：~~无法确定占10%？~~。模型融合。伪标签。改进分词。~~截取q变长~~。

21：在14保留Exact match和option match和cos sim的基础上，class weight对1类权重为2，Val极差。class weight对0类权重为2，val也差。不可行。

22：和BIDAF3保留exact match和option match和cos sim参数相同。删除最后的query aware representation的self-attention，val=0.8129，反而变好。增加词数到300000，val=0.8093，变差 。q=15，val=0.8109，作用较小。

23：BIDAF5：采用以上所有最强参数。第6代Val=0.8122。调整reduceLRonPlateau基于val loss，第5代val=0.8130。若rnn=64，第7代val=0.8120。回退到22：删除最后的query aware representation的self-attention，val=0.8114<0.8129无法复现？操！那我优化个叽霸！

在rnn=64基础上加入u经过LSTM的表示um，val=0.8141。再加u，或全连接层都没有更好。

尼玛突然发现jieba分词表现的和没加词典一样？why？fuck！！！好像train的分多了，词典有变，影响到了valid的分词结果。数字分错，歧义不行，‘能’分错。怎么解决？

每一条动态加词典，分词后立即恢复：数字能分开，歧义不行，‘能’看情况。

尝试直接原装分词不加词典：数字能分开，歧义不行，‘能’能分开。曹尼玛费了半天劲，不如不加词典！！！在rnn=64加um的基础上，val=0.8194。直接把选项放问题开始，val=0.8203，更好。Rnn=128，val=0.8198。加一层dense，val=0.8194.不好。改回rnn=64。

把搜狗词库加入jieba词典。Val=0.8173.不好。

24：match-LSTM0：简化的r-net。参数和BIDAF5的rnn=64加um相同。Val=0.8165<.08194略差于BIDAF。

未来改进点：模型融合（4种模型，pre or post）。伪标签和验证集参与融合。改进训练策略，3代后放开词向量训练。参考 <https://zhuanlan.zhihu.com/p/45391378>

以下采用新分词和问题数据。

25：BIDAF6：采用以上所有最强参数。尝试精确训练策略。Rnn=64。

a. Lr=0.001，test\_feature\_p, test\_feature\_q，1代0.8222，第2代变差。lr/2，2代都变差。lr/2，内存溢出。总共4代就超越0.8203。

b. 如果先练4代0.8186。（a）lr/5，放开词向量，2代0.8208，第3代变差。（b）lr/10，放开词向量，1代0.8210。

rnn=128。最强BIDAF单模。保存在my6.h5。（a）Lr=0.001，先练3代0.8174。lr/5，放开词向量，1代0.8224，第2代变差。Lr/2，更差。（b）Lr=0.001，先练2代0.8132。lr/5，放开词向量，1代0.8207，第2代变差。比rnn64提高不明显。

Rnn=64,4代val=0.8213。

Q也加上特征后，同上，0.001先练4代0.8193，lr/5，放开词向量1代0.8235。test6=0.7349。

改用preprocess2的文本，同上，0.001先练4代0.8202，lr/5，放开词向量1代0.8245。test6=0.7357。果然有提高。无法确定占比0.055

计算similarity matrix时使用scaled dot product attention。0.001先练4代0.8220，lr/5，放开词向量1代0.8231<0.8245。不好。

Shuffle=False，val=0.8218.不好。

26：BIMPM0：删除了max attentive matching（因为keras无法实现）只剩3个strategy。Max feature=120000, q=10, batch=128, rnn=32, mp=3, dense=200，因为太慢所以全面缩水模型参数。0.001训练3代后放开词向量0.0002训练1代val=0.8138 。test2=

27：HAFN0：rnn=128。最后加入um。0.001训练3代后放开词向量0.0002训练1代val=0.8183。Q也加上特征后，rnn=128，0.001先练3代0.8187，lr/10，放开词向量3代0.8202。test4=

改用preprocess2的文本，同上，0.001先练4代0.8217，lr/5，放开词向量1代0.8198。重试：0.001先练3代0.8155，lr/5，放开词向量2代0.8207。test4=0.7259。无法确定占比0.0471

28：rnet0：rnn=128。最后加入um。0.001训练3代后放开词向量0.0002训练2代val=0.8175。

Q也加上特征后，同上，0.001先练3代，lr/5，放开词向量2代0.8204。Test1=0.7308。无法确定占比0.067

29：match-LSTM0：rnn=128。最后加入um。0.001训练3代后放开词向量0.0002训练1代val=0.8143。

Q也加上特征后，同上，0.001先练3代0.8181，lr/10，放开词向量反而不如。直接训练到5代0.8194。test3=0.73无法确定占比0.06

30：rcnn0：rnn=128。P1,3,5，q1,3。Filter=128。第5代最高，Val=0.8207。超过BIDAF！

Q也加上特征后，同上，0.001先练5代0.8203，lr/5，放开词向量1代0.8220。再次尝试，0.001先练4代0.8215，lr/10，放开词向量1代0.8230。test5=

若用Siamese结构，7代val=0.8209没进步。

改用preprocess2的文本，同上，0.001先练4代0.8207，lr/5，放开词向量1代0.8241。test5=0.7304。无法确定占比0.0437。testb5=0.6118。只用train做embedding，val=0.8215，testb=0.6938

融合上面6个模型：1-6 test=0.7471。去除最差的2，test=0.7488。只有最好的两个（5+6），test=0.7432。

未来改进点：~~换用HanLP分词的NLP tokenizer~~？知识图谱？更多文本相似度模型可以移植。~~词性标注~~？~~同义词扩展~~？ELMo and BERT。~~Scaled dot product attention~~。

同义词扩展太长，passage扩到1000个词以上。放弃。

31：Rcnn-att0 增加了feed forward self attention，val=0.8233。比rcnn略强。Test7=0.7294。三合一后的表现不如BIDAF。无法确定占比0.0523

用preprocess2的文本，同上，0.001先练4代0.8217，加入伪标签，lr/5，放开词向量1代0.8235。

不用伪标签更好，4代val=0.8241。test7=。无法确定占比0.0388

32：BIDAF-att0：22删除了self-attention，根据论文改进后再次尝试。在BIDAF6，preprocess2的基础上0.001先练4代0.8205，lr/5，放开词向量1代0.8227<0.8245。还是不好。放弃。

33：BIDAF-ELMo0：BIDAF6的input和rnn output上加ELMo，IO太慢，5小时一代。放弃。

把ELMo的output放到C盘SSD上。1小时一代，val=0.8131，变差。

缩放系数0.01，val=0.8124，更差。

34：rcnn1，在30的基础上全面增强，p的filter和rnn \*2。q1,2,3，增加一层dense，换GRU。Val=0.8238。无效。

35：BIDAF7: 用preprocess\_short处理的3合1文本，softmax。12代val=0.6776，放弃。

36：BIDAF8：将valid set和train的前90000条数据交换，并在preprocess4中交换p和q的pre和post，再训练BIDAF6。

4代0.8167后，加入伪标签，再从头训练。0.001练4代0.8197，重新预测伪标签，lr/5，放开词向量1代val=0.8217。又不灵了。放弃。

0.001先练3代0.8181，lr/5，放开词向量1代val=0.8231。此时再加入伪标签，从头训练，0.001练4代0.8197， lr/5，放开词向量1代val=0.8231。还是没提高，回滚到上一个0.8231。test8=。无法确定占比0.0479

37：BIDAF9：BIDAF6基础上伪标签。0.001先练4代0.8193，lr/5，放开词向量，加入伪标签1代val=0.8235<0.8245。无效。

4代0.8193后，加入伪标签，再从头训练。0.001练3代0.8207，重新预测伪标签（因为kernel died），lr/5，放开词向量1代val=0.8259。提高！test9=0.7382。无法确定占比0.0436。将无法确定的概率调成2倍，占比0.0934，test=0.7291。简单提高概率不好。

38：在BIMPM0的基础上Max feature=150000, q=15, batch=64, rnn=64, mp=3, dense=200，加入feature。40分钟一代，5代val=0.8151.不能用。

39：BIDAF10：在BIDAF6的基础上val=0.8231，加手工特征（preprocess3）。

对p加窗口Jaccard相似度，归一化编辑距离。都是基于窗口内的词。把截断放到提取特征之后，保留更多信息。Val=0.8222。变差。

只有基于词的归一化编辑距离，val=0.8222。一样差。改成基于字，val=0.8219，更差。

只有基于词的窗口Jaccard相似度，val=0.8216，更差。改成基于字，val=0.8232，持平。

对p和q加词向量广义Jaccard相似度，val=0.8201。变差。

窗口fasttext相似度。Val=0.8070。不能用。

加入观点词库特征（fea2）。Val=0.8217。词更少，val=0.8211。词更多，val=0.8070，更差。

加character match一个字是否在另一句出现。Val=0.8224变差。

在preprocess1里用HanLP的nshort分词同时加入5维pos-tag特征（tsv1，fea3），val=0.8218。换用NLPtokenizer分词效果好，但是连空格都不单独分！先split再分词可以避免，Val=0.8206。

由此联想到preprocess2里用jieba分词有的符号仍然保留。换用pseg去除x和w符号。48547全公式出错，放弃。

40：preprocess5，（tsv2，fea4）句子平均词向量相似度对比。78954-78956全英文出错，改掉。只取p中最相关的一句，BIDAF10，val很差。保留p中距离少于0.3的，放弃。

41：Preprocess6，先分出可回答和不可回答的。第一轮数据集na，test不带answerable，1个手工特征（em）fea5。用BIDAF11继承于BIDAF10，预测可回答的test，保存在test\_na\_pred.csv。再回到preprocess6，按照preprocess2的方法，train和valid只处理可回答的问题，test仍预测全部。第二轮数据集a。2个手工特征fea6。用BIDAF11，val=0.7798。判断：正反答案相差很大的（大于0.7），一般都不会无法确定，选较大的。然后选择无法确定的（na小于0.5）。最后选择较大的。Test11=0.7049。 检查发现无法确定占比0.0213。

Stage2：

Preprocess2预处理test数据，保存在output。Train和valid预处理的数据和特征直接上传到work。

1：rnet0训练0.8187，0.8202，rnet01预测

2：rcnnatt0训练0.8265，rcnnatt01预测

3：match0训练0.8205，match01预测

4：HAFN0训练0.8175，0.8202，HAFN01预测

5：rcnn0训练0.8235，rcnn01预测0.6938

6：BIDAF6训练0.8231，BIDAF61预测

1-6集成0.6675。改变无法确定的threshold，0.6941

Preprocess4改变train和valid截取

BIDAF8使用valid训练，改变截取。

始终比stage1低5个点！改成在虚拟机上训练。为了节约时间，batch=512

1：rnet0训练，rnn=64，360s，0.8194

2：BIDAF6训练，rnn=64，270s，0.8240

3：match0训练，rnn=64，260s，0.8201

4：HAFN0训练，rnn=64，350s，0.8164

拆成train和val两个脚本，val用valid数据替换等量的train数据，并交换pad的方向。

为了限制在2小时内，改GRU，反而更好。

Train.py

1：match0训练，rnn=64，260s，0.8171，改成直接训练5代。

2：BIDAF6训练，rnn=64，270s，0.8243

Val.py

3：match0训练，rnn=64，260s，

4：BIDAF6训练，rnn=64，270s，

Train.py结束总会有segmentation fault，core dump。改回batch=256，舍弃3。仍有此错误，无奈到时提交。