



# Sentiment Classification based on Simplified Chinese News Text

2020年5月

汇报人：徐子康

# 实验背景：

情感分析(Sentiment Analysis)是帮助用户快速获取、整理和分析相关评价信息，对带有情感色彩的主观性文本进行分析、处理、归纳和处理。

本实验聚集于文本主观倾向性分类，所研究的对象是文本的“主观因素”，即作者表达出来的主观倾向性，分类的结果是对于一个特定的文本要得到它是否支持某种观点的信息。这种倾向性情感文本分类任务又被称为情感分类(Sentiment Classification)

# 实验数据：

实验数据来源于新浪微博收集的4570篇中文新闻文章。每条样本中包含了不同数量的用户的情感打分，共8种情感类别，每条样本至少含有6个单词和1个用户评分。数据已通过中文分词处理，每行数据由时间戳，情感分布，文本三个部分组成，以tab(\t)分割。

使用2012年1月至2月发布的2342篇新闻文本构建训练集，使用2012年3月至4月发布的2228篇新闻文本构建测试集

# 实验摘要：

本次实验包含了CNN，RNN与MLP的多种模型基于PyTorch所实现的Baseline，使用Tensorboard进行可视化分析。

因为硬件的限制，加之数据集不是那么优秀，实验未能取得最好的结果，下面将会具体阐述。

# 实验说明&如何运行本实验



- 项目代码地址：[https://github.com/mrxgavin/Sinanews\\_SentimentClassification](https://github.com/mrxgavin/Sinanews_SentimentClassification)
- code文件夹存放代码，具体请参照README.md 文件
- 为了便于实验复现，code文件夹中，configs文件夹存放了模型配置文件，模型配置说明可以参考./code/config\_description.md 文件。输入' python main.py --config [config\_path] '即可按照指定配置文件进行训练。或者也可以运行./code/runall.sh 文件
- main.py: 程序的主入口，主要用来加载文件，调用数据集、模型和训练的接口，在流程中传递数据。
- dataset.py: 加载数据集的接口，使用了torchtext将数据分成 Index、Label、Text三个 Field，通过其前后处理的接口将数据去字母和数字，将次数最多的Label作为最好的Label，还进行了混洗等操作。
- model.py: 定义了模型结构，具体结构解释见后面内容。
- trainer.py: 训练、测试，可视化。

# 模型网络结构-CNN

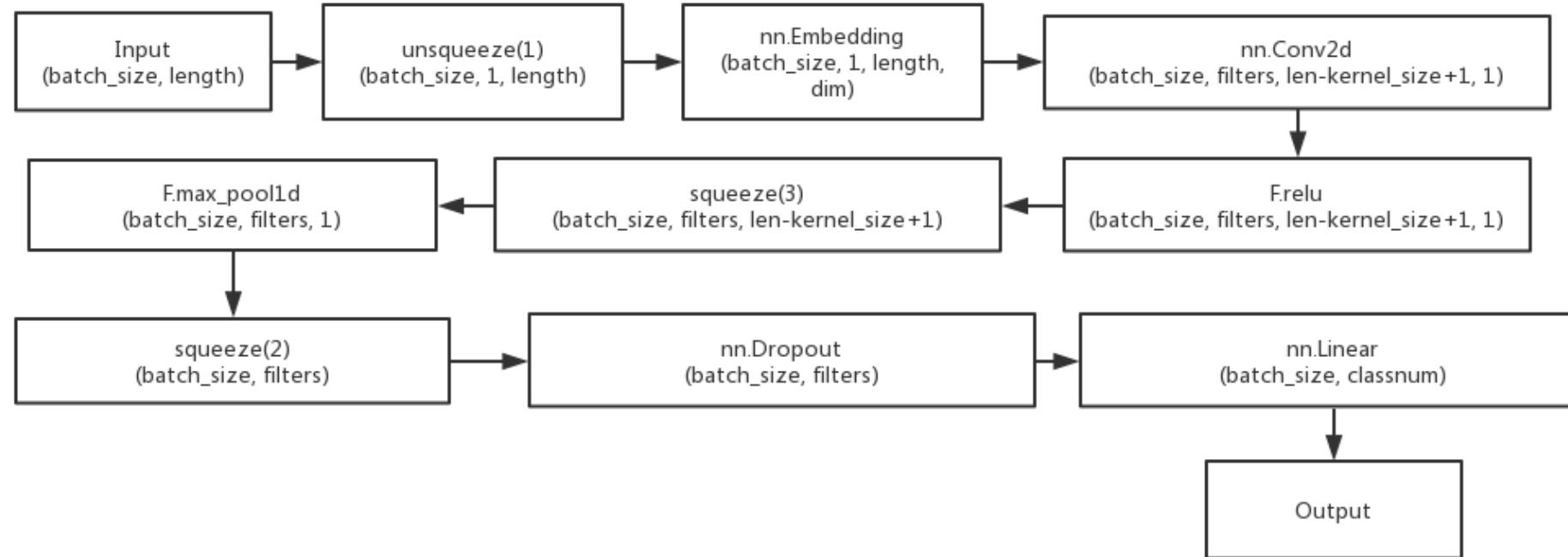


图1. CNN 网络结构

参考论文实现了代码，实现了论文中的：**A.CNN-rand**(All words are randomly initialized and then modified during training); **B.CNN-static**(Use pre-traind vectors from word2vec, all words are kept static and only the other parameters of the model are learned ); **C.CNN-non-static**(The same as CNN-static but the pretrained vectors are fine-tuned for each task)

Reference: Kim, Y., 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

# 模型网络结构-RNN % MLP

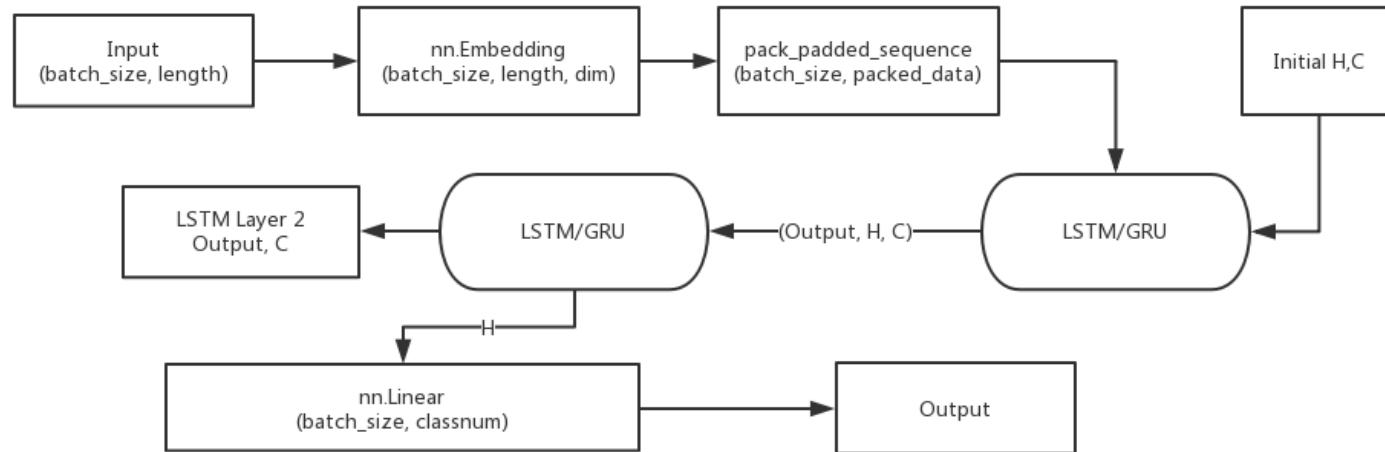


图2. RNN 网络结构

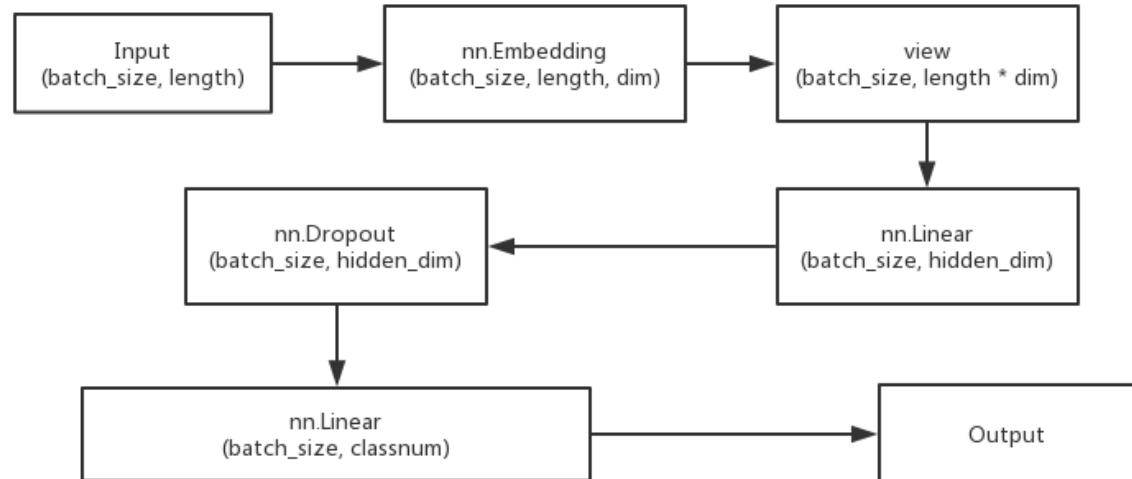
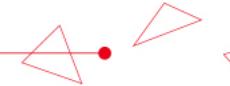


图3. MLP 网络结构



- 配置文件中提供了Loss函数的选择，包含交叉熵（cross entropy）或均方误差（MSE）

## Cross Entropy:

pytorch中的nn.CrossEntropyLoss已经顺序包含了nn.softmax, nn.Log & nn.NLLLoss, 把网络的最后一层输出映射到[0,1]区间的概率，之后取对数计算交叉熵作为Loss。

## MSE:

在数据处理时保留情感分布而非最大值，因此在得到最后一层后在经过一次softmax得到概率分布后，直接和原始数据分布计算MSE作为Loss。

# MSE作为Loss函数

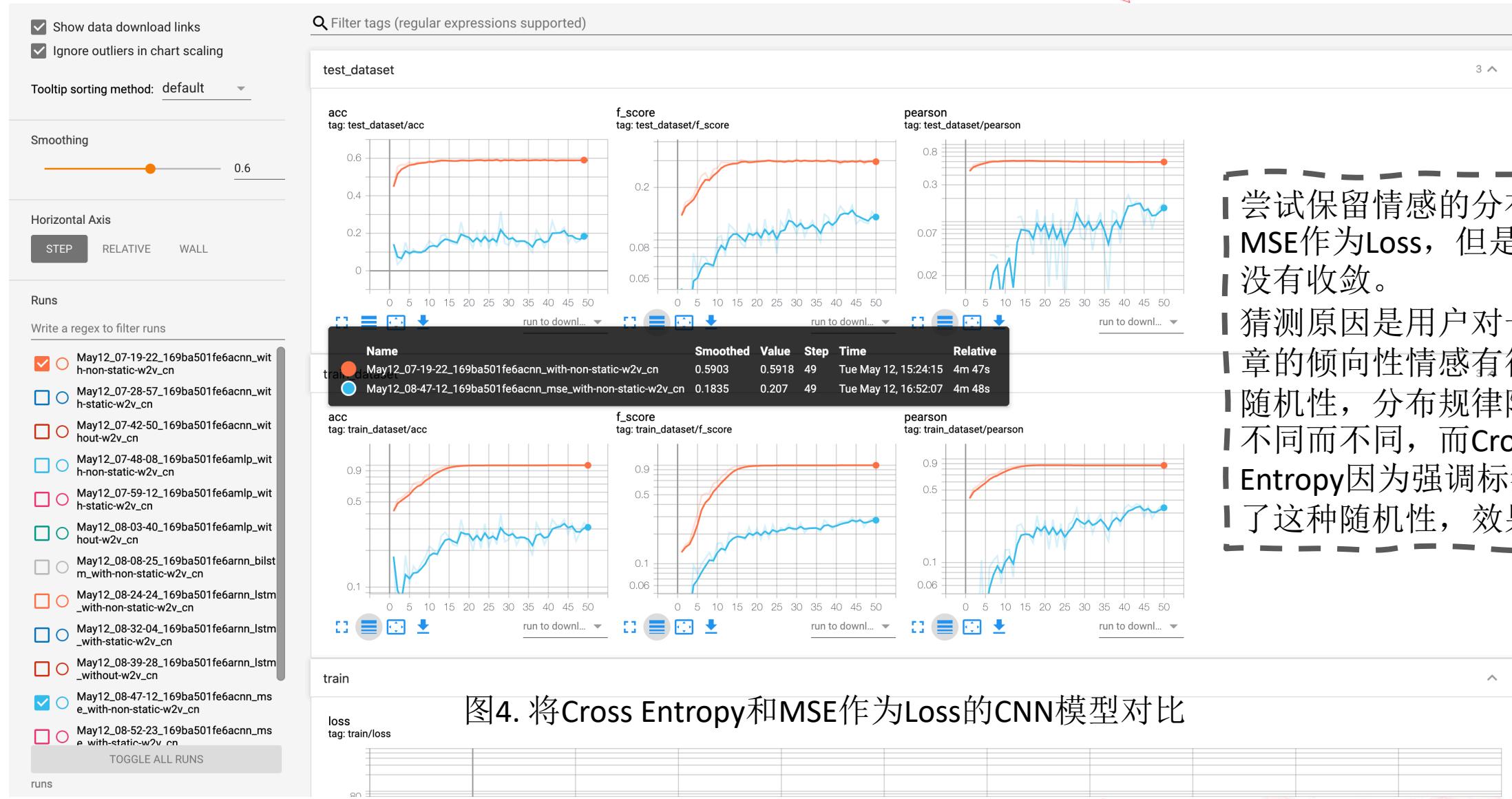


图4. 将Cross Entropy和MSE作为Loss的CNN模型对比

尝试保留情感的分布并将MSE作为Loss，但是MSE并没有收敛。  
猜测原因是用户对一篇文章的倾向性情感有很大的随机性，分布规律随时间不同而不同，而Cross Entropy因为强调标签忽略了这种随机性，效果更好。

Show data download links  
 Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing: 0.6

Horizontal Axis: STEP RELATIVE WALL

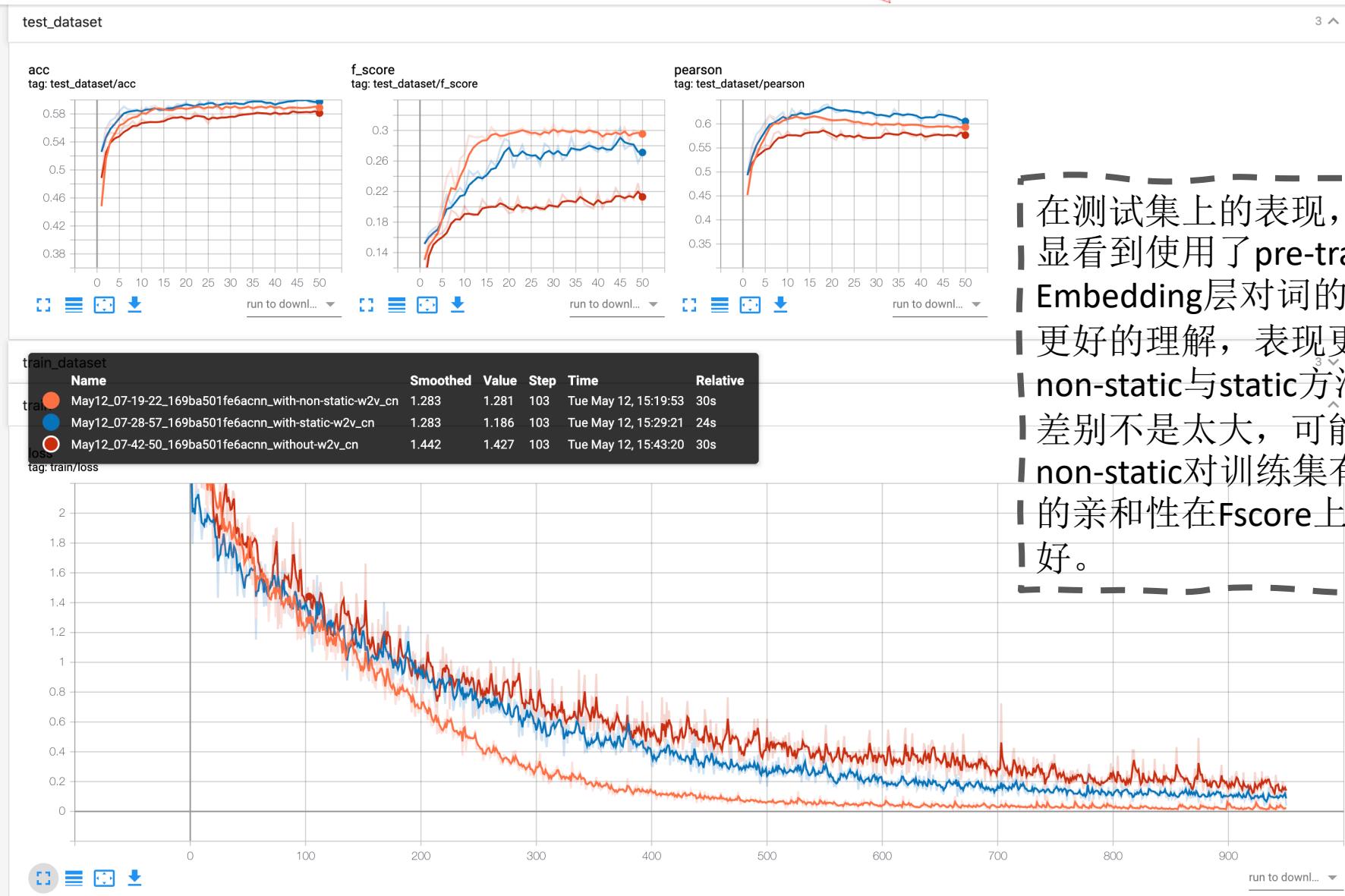
Runs

Write a regex to filter runs

- May12\_07-19-22\_169ba501fe6acnn\_wit h-non-static-w2v\_cn
- May12\_07-28-57\_169ba501fe6acnn\_wit h-static-w2v\_cn
- May12\_07-42-50\_169ba501fe6acnn\_wit hout-w2v\_cn
- May12\_07-48-08\_169ba501fe6amlp\_wit h-non-static-w2v\_cn
- May12\_07-59-12\_169ba501fe6amlp\_wit h-static-w2v\_cn
- May12\_08-03-40\_169ba501fe6amlp\_wit hout-w2v\_cn
- May12\_08-08-25\_169ba501fe6arnn\_bilst m\_with-non-static-w2v\_cn
- May12\_08-24-24\_169ba501fe6arnn\_lstm \_with-non-static-w2v\_cn
- May12\_08-32-04\_169ba501fe6arnn\_lstm \_with-static-w2v\_cn
- May12\_08-39-28\_169ba501fe6arnn\_lstm \_without-w2v\_cn
- May12\_08-47-12\_169ba501fe6acnn\_ms e\_with-non-static-w2v\_cn
- May12\_08-52-23\_169ba501fe6acnn\_ms e\_with-static-w2v\_cn

TOGGLE ALL RUNS

runs



在测试集上的表现，能明显看到使用了pre-trained的Embedding层对词的语义有更好的理解，表现更佳。  
non-static与static方法比较，差别不是太大，可能因为non-static对训练集有更好的亲和性在Fscore上表现更好。

# RNN-LSTM

Show data download links  
 Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing: 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

- May12\_07-42-50\_169ba501fe6arnn\_wit hout-w2v\_cn
- May12\_07-48-08\_169ba501fe6amplp\_wit h-non-static-w2v\_cn
- May12\_07-59-12\_169ba501fe6amplp\_wit h-static-w2v\_cn
- May12\_08-03-40\_169ba501fe6amplp\_wit hout-w2v\_cn
- May12\_08-08-25\_169ba501fe6arnn\_bilst m\_with-non-static-w2v\_cn
- May12\_08-24-169ba501fe6arnn\_lstm \_with-non-static-w2v\_cn
- May12\_08-32-04\_169ba501fe6arnn\_lstm \_with-static-w2v\_cn
- May12\_08-39-28\_169ba501fe6arnn\_lstm \_without-w2v\_cn
- May12\_08-47-12\_169ba501fe6acnn\_ms e\_with-non-static-w2v\_cn
- May12\_08-52-23\_169ba501fe6acnn\_ms e\_with-static-w2v\_cn
- May12\_09-17-42\_169ba501fe6arnn\_gru \_with-non-static-w2v\_cn
- May12\_09-25-52\_169ba501fe6arnn\_gru \_with-static-w2v\_cn

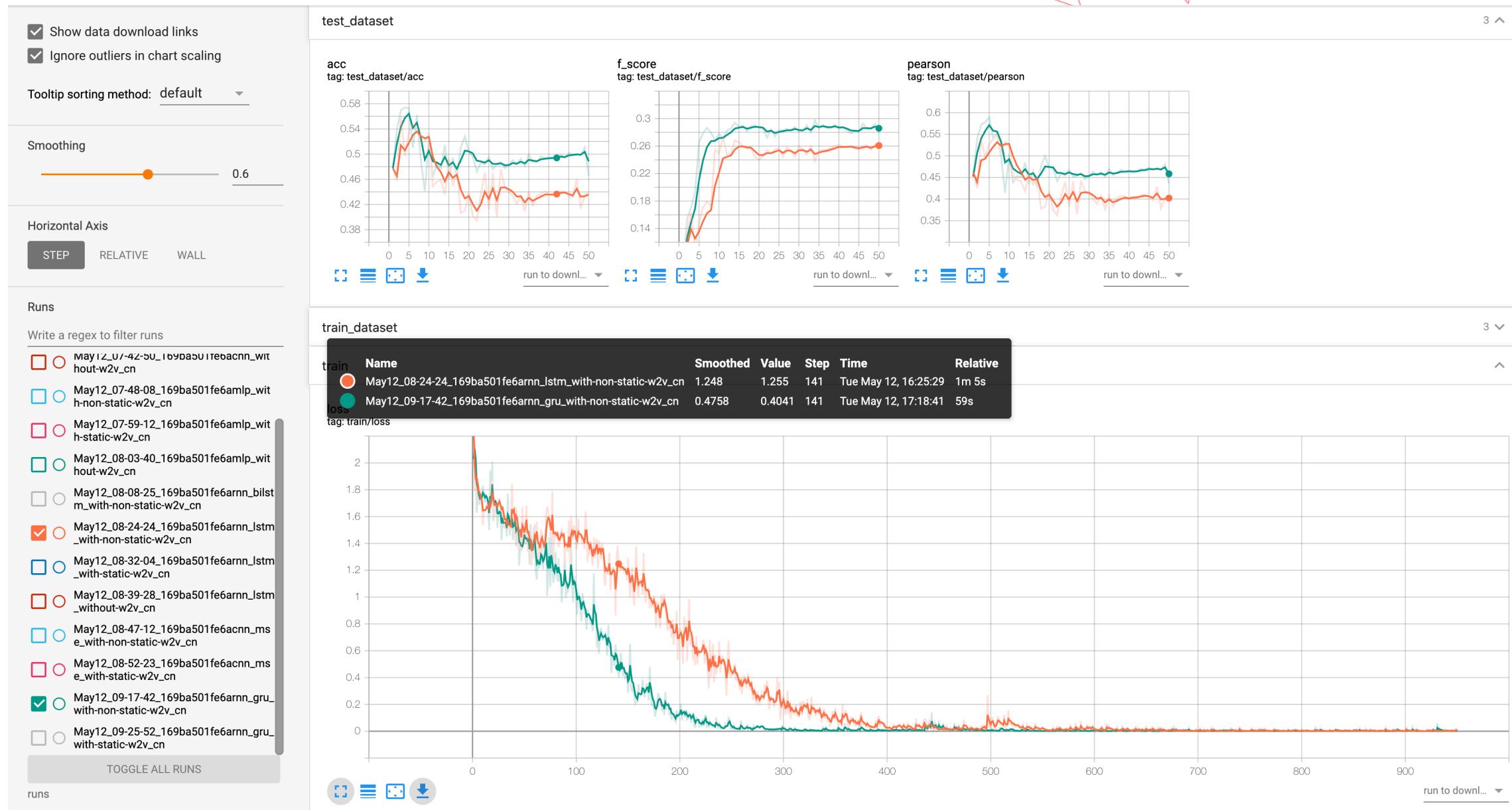
TOGGLE ALL RUNS

runs



RNN出现了一定程度的过拟合，acc后面波动显著，Fscore一直在增加，相关性系数也存在一定程度的过拟合。

# RNN-GRU & LSTM



# RNN-Bidirectional LSTM

Show data download links  
 Ignore outliers in chart scaling

Tooltip sorting method: default ▾

Smoothing: 0.6

Horizontal Axis: STEP RELATIVE WALL

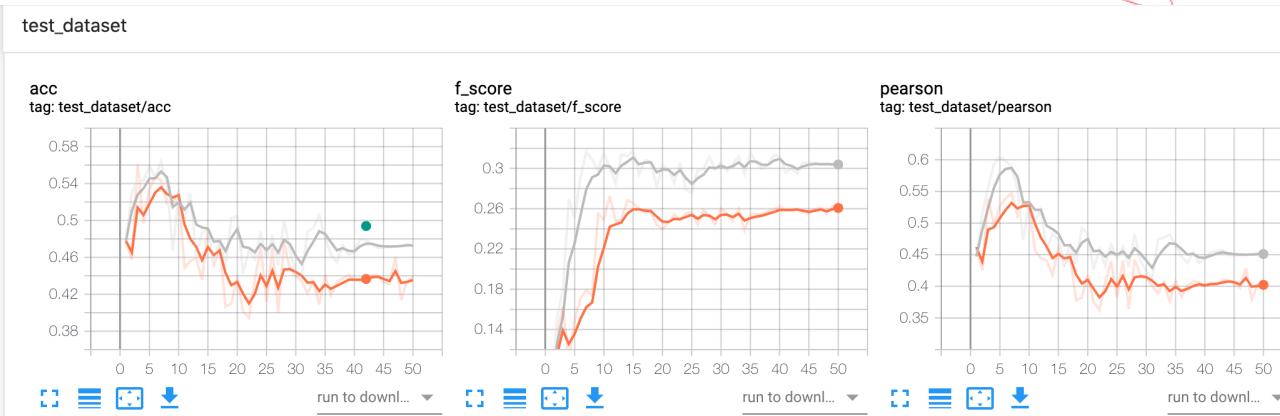
Runs

Write a regex to filter runs

- May12\_07-19-22\_169ba501fe6acnn\_wit\_h-non-static-w2v\_cn
- May12\_07-28-57\_169ba501fe6acnn\_wit\_h-static-w2v\_cn
- May12\_07-42-50\_169ba501fe6acnn\_wit\_hout-w2v\_cn
- May12\_07-48-08\_169ba501fe6amplp\_wit\_h-non-static-w2v\_cn
- May12\_07-59-12\_169ba501fe6amplp\_wit\_h-static-w2v\_cn
- May12\_08-03-40\_169ba501fe6amplp\_wit\_hout-w2v\_cn
- May12\_08-08-25\_169ba501fe6arnn\_bilstm\_with-non-static-w2v\_cn
- May12\_08-24-24\_169ba501fe6arnn\_lstm\_with-non-static-w2v\_cn
- May12\_08-32-04\_169ba501fe6arnn\_lstm\_with-static-w2v\_cn
- May12\_08-39-28\_169ba501fe6arnn\_lstm\_without-w2v\_cn
- May12\_08-47-12\_169ba501fe6acnn\_ms\_e\_with-non-static-w2v\_cn
- May12\_08-52-23\_169ba501fe6acnn\_ms\_e\_with-static-w2v\_cn

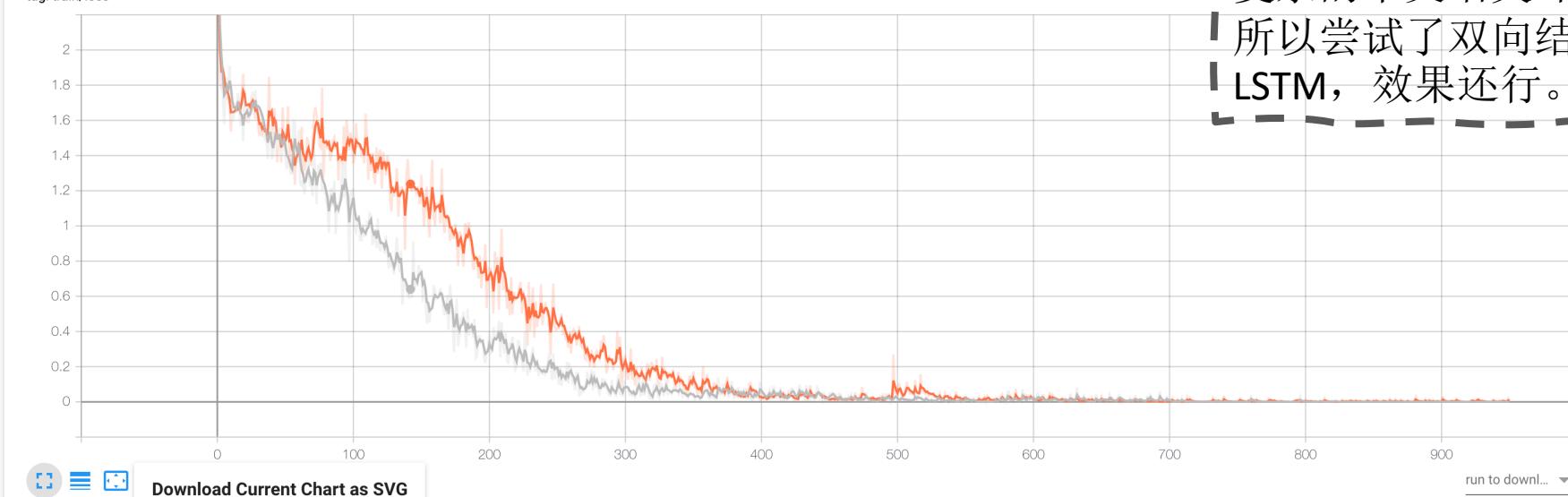
TOGGLE ALL RUNS

runs



### train\_dataset

train	Name	Smoothed	Value	Step	Time	Relative
train	May12_08-08-25_169ba501fe6arnn_bilstm_with-non-static-w2v_cn	0.6401	0.6643	142	Tue May 12, 16:10:41	2m 15s
loss	May12_08-24-24_169ba501fe6arnn_lstm_with-non-static-w2v_cn	1.238	1.224	142	Tue May 12, 16:25:30	1m 5s



单向的RNN只能对输入的历史状态有记忆功能，但是不能对未来的上下文信息作出一定的判断，尤其是在复杂的中文语义环境中。所以尝试了双向结构的LSTM，效果还行。

Show data download links  
 Ignore outliers in chart scaling  
 Tooltip sorting method: default

Smoothing:

Horizontal Axis: STEP, RELATIVE, WALL

Runs: Write a regex to filter runs

- May12\_07-19-22\_169ba501fe6acnn\_wit\_h-non-static-w2v\_cn
- May12\_07-28-57\_169ba501fe6acnn\_wit\_h-static-w2v\_cn
- May12\_07-42-50\_169ba501fe6acnn\_wit\_hout-w2v\_cn
- May12\_07-48-08\_169ba501fe6amplp\_wit\_h-non-static-w2v\_cn
- May12\_07-59-12\_169ba501fe6amplp\_wit\_h-static-w2v\_cn
- May12\_08-03-40\_169ba501fe6amplp\_wit\_hout-w2v\_cn
- May12\_08-09-25\_169ba501fe6arnn\_bilstm\_with-non-static-w2v\_cn
- May12\_08-24-24\_169ba501fe6arnn\_lstm\_with-non-static-w2v\_cn
- May12\_08-32-04\_169ba501fe6arnn\_lstm\_with-static-w2v\_cn
- May12\_08-39-28\_169ba501fe6arnn\_lstm\_without-w2v\_cn
- May12\_08-47-12\_169ba501fe6acnn\_ms\_e\_with-non-static-w2v\_cn
- May12\_08-52-23\_169ba501fe6acnn\_ms\_e\_with-static-w2v\_cn

TOGGLE ALL RUNS

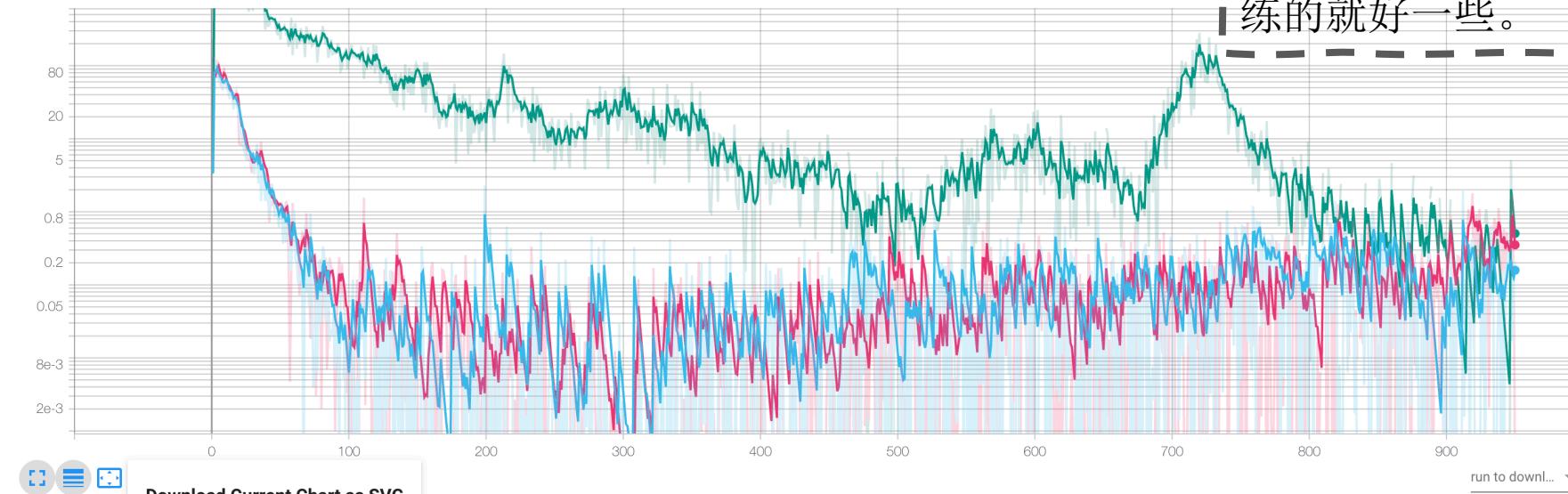
runs



train\_dataset

train

loss tag: train/loss



MLP因为网络结构简单所以训练速度很快，但是没有加预训练模型的MLP训练过程非常抖动，推测和其两层线性结构有关，在Embedding层的巨大变化可能导致对结果的影响较大。加了预训练的就好一些。

## CNN

优点是速度非常快，而且效果还行。我对于CNN的理解是通过卷积来对每kernel\_size个词语进行特征提取，同时这些特征也通过max\_pool和relu来激活，最后把特征通过来映射到分类的结果上。缺点是，在一定程度上“记忆”短暂，如果在非常复杂的语义环境下，可能会出现断章取义的情况。

## RNN

相对于迭代次数而言，RNN是学习得最快的，在前几个Epoch就达到了非常高的准确率，其中的长期记忆的设计真正的在一定程度上避免了断章取义的问题，实现了信息的持久化。就结果上看，RNN比较容易出现过拟合，猜测是RNN的Cell中的tanh导致的。

## MLP

MLP是结构最简单的，在本实验的三个模型中，同时训练也是最快的，不容易过拟合。我对于MLP的理解是其不会记住特征，但是却一次性充分利用了所有的信息，不足的是，结构的过于简单导致了断章取义的现象。从结构和实验结果来看，MLP是一个很好的方法和思想。