2017 知乎看山杯 ye 组参赛方案说明

黄永业
北京邮电大学
yongye@bupt.edu.cn

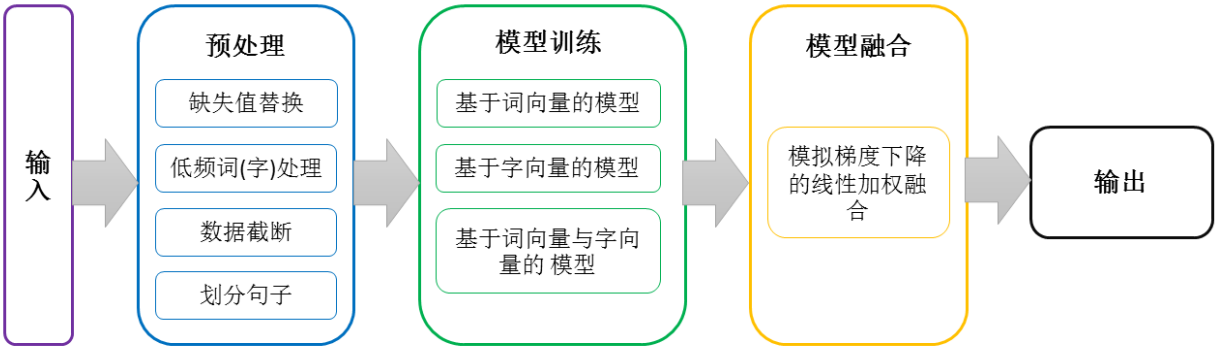

图 1. 方法流程图

摘要

在本次比赛中，知乎给出了问题与话题标签的绑定关系的训练数据，通过这些数据训练出能够对未标注的数据自动标注的模型。我们组主要利用深度学习的方法，以文本分类中比较经典的双端 GRU 模型和 TextCNN 等模型为基础，构造多个新的模型进行分类。然后模拟梯度下降的方法对多个模型进行线性加权融合，得到最优的结果。通过本方法，本组最终在 Public 排行榜上得分为 0.43296，排名第五；在最终得分榜上得分 0.43060，排名第六。

关键词 知乎看山杯，文本分类，深度学习，模型融合

1 概述

我们的方法主要包括下面四个步骤：数据预处理，特征提取，模型训练和模型融合。

在数据预处理中，完成了缺失值处理、低频词(字)处理、数据截断和划分句子等处理。在特征提取中，我们主要是利用了赛方提供的词向量和字向量两部分特征，没有引入更多的特征。在模型训练部分，根据输入数据的不同，主要训练了三大类型的模型：仅使用词向量的模型，仅使用字向量的模型，同时使用词向量和字向量的模型。在模型融合部分，我们模拟梯度下降的方法进行多个模型的线性加权融合，利用线下验证集的 F1 值变化来调整各个模型的权重。

1.1 实验流程

图 1 展示了我们方法的主要流程。

2 数据预处理

2.1 缺失值处理

赛方提供了训练集和测试集两个数据集，其中训练集包含 2999967 个问题，测试集包含 217360 个问题。每个问题由问题标题和问题描述两部分组成。在两个数据集中，都有部分问题缺失标题或者缺失描述，表 1 统计了两个数据集中存在缺失值的问题数量（基于词进行统计）。

表 1. 缺失数据统计表

数据集	标题缺失	描述缺失
训练集	15	834804
测试集	3	60234

在测试集中，我们把缺失的标题用该问题的描述进行填充，同理，缺失的描述利用对应问题的标题进行填充。在训练集中，处理方式和对测试集的处理基本相同，只是对于没有标题的 15 个问题，我们直接丢弃，这样最后用于训练和验证的样本数量为 2999952(2999967-15)个。

2.2 低频词(字)处理

赛方提供了字符级别的 256 维的 embedding 向量及词语级别的 256 维的 embedding 向量。但是词汇表中省略掉了出现频次为 5 以下的字符或者词语，因此在训练和验证语料中出现的词汇有可能没有对应的 word embedding 向量。对于没有出现在词汇表中的词或字，我们统一给定一个随机初始化的向量来表示。

2.3 数据截断

由于每个问题的标题和描述长度不一，为了方便模型的训练，我们将所有样本的数据都截成统一的长度。在词汇级别上，每个问题的标题截取 30 个词（部分模型截取 50 个词），长度少于 30 个词的数据使用'UNKNOWN'进行填充。对于问题描述和在字级别上，使用同样的方法进行处理，最后每部分截取的长度如表 2 所示。

表 2. 截取长度

	标题长度	描述长度
词数	30/50	150
字数	52	300

2.4 划分句子

划分句子主要是为了训练分层模型。分层模型的主要思想是：每个文档是由多个句子构成的，每个句子又是由多个词(字)构成的。普通模型都是直接对整个文档进行处理，而分层模型一样先处理每个句子，得到每个句子的表示向量，然后再用这些句子的表示向量来学习整个文档的信息。这种处理方式对于较长的文档效果更加明显，而且对于 RNN 模型还能够极大地降低时间复杂度。

由于问题的标题部分一般较短，而问题描述部分一般较长，所以我们主要对描述部分进行划分。通过分析，我们发现问号，句号和感叹号对应的编码应该是：'w111', 'w23', 'w1570'。每个问题描述一共截断为 10 个句子，每个句子截断为 30 个词。对于字级别使用同样的方式进行处理，每个问题描述一共截断为 10 个句子，每个句子截断为 52 个字。

3 特征提取

在所有的模型中，我们只使用了赛方提供的词向量和字向量作为初始特征，没有使用其他统计信息。

4 模型说明

根据输入的不同，我们主要训练了三大类型的模型：仅使用词向量的模型，仅使用字向量的模型，同时使用词向量和字向量的模型。

对于仅使用词向量或者仅使用字向量的模型，我们将问题标题和问题描述两部分分层两个通道，分别使用 Bi-GRU(双端 GRU)模型或者 TextCNN 模型进行训练，各自得到一个表示向量，然后将两个向量通过拼接或者按元素取最大值的方式合成一个向量，最后通过全连接层和 sigmoid 层输出每个类的预测概率。对于同时使用词向量和字向量的模型，相当于每个模型有四个通道输入。

通过实验发现，仅使用词向量的单模型和同时使用词向量与字向量的单模型表现要比仅使用字向量的单模型好很多。但是在最后的模型融合中，加入仅使用字向量的模型能够有效地提高融合的性能。此外，为了提高融合的性能，在训练的过程中，应当尽量的增加不同模型之间的差异性。比如使用不同的输入(词、字)，在 Bi-GRU 中设置不同的隐藏节点数，在 TextCNN 中使用不同的核大小与核数量，使用不同的初始化方式，是否使用 dropout 层等。尽管有些单模型性能一般，但是由于与其他模型差异较大，最后能够有效地提高模型融合的性能。下面介绍我们用到的几个主要的模型结构(词和字级别的模型结构相似)。

4.1 TextCNN

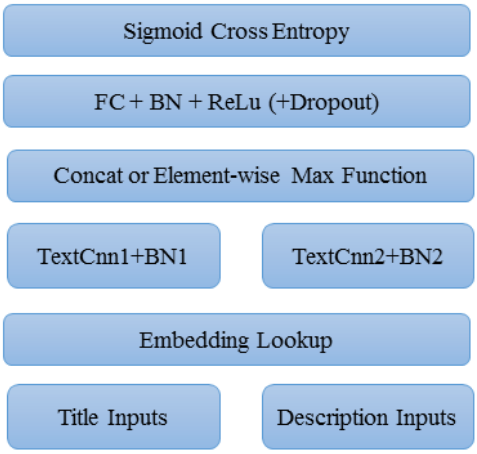

图 2. TextCNN 网络结构

4.2 Bi-GRU

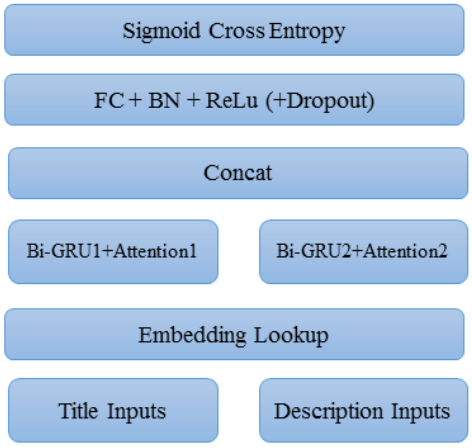

图 3. Bi-GRU 网络结构

4.3 Hierarchical Model

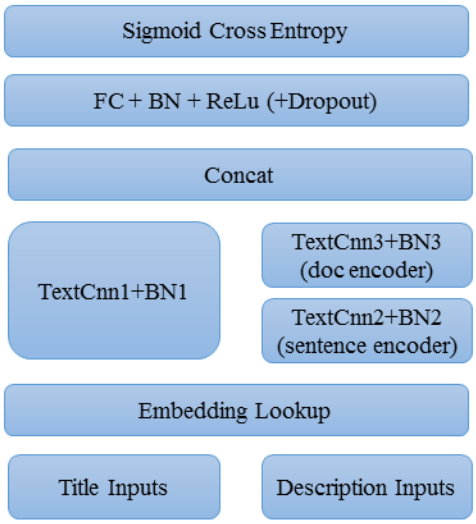

图 4. Hierarchical TextCNN 网络结构

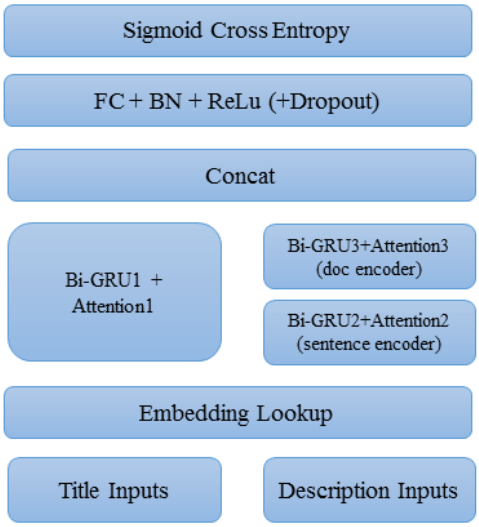

图 5. Hierarchical Bi-GRU 网络结构

4.4 Multi-Embedding Model

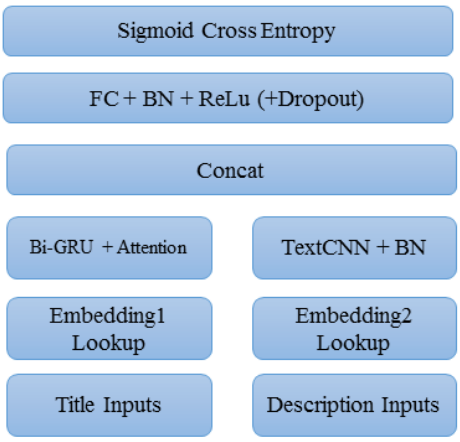

图 6. Multi-Embedding 网络结构

5 模型融合

在最终提交的结果中，我们一共融合了 30 多个不同的模型，在第 4 部分我们已经介绍了过不同模型的结构。这些模型之间性能差异较大，如果采用直接平均加权的话，融合效果较差。我们主要尝试过两种融合方式。

第一种是使用归并的方式将模型两两融合，通过线下验证集的 F1 值变化寻找两个模型最佳的权重比例。然后使用同样的方式对融合后得到的模型继续融合。但是实验发现这种方式效果很不理想，甚至比不上简单的手动调整权重。

第二种是模拟梯度下降的方式来进行融合，这种方式效果明显好于归并方式和手动调整，但是可能会对验证集过拟合，所以还需要通过线上测试集的最终表现来判断融合的效果。我们以固定的随机方式将训练集中的样本打乱，然后取固定的 100000 个样本作为线下验证集。对于每个训练好的单模型，计算验证集中每个样本属于各个样本的概率，则得到一个 100000×1999 的预测概率矩阵。假设有 K 个单模型，每个单模型的输出为 $S_i \in R^{100000\times1999}$, $i = 1,2,\dots,K$. $score_eval(S)$函数返回在验证集上预测的 F1 值，然后按照下面的方式来进行权重的调整：

步骤 1：根据各个单模型的性能，手动给每个单模型赋予初始权重，计算融合后验证集的 F1 值。

$$S = \sum_{1}^{K} w_i S_i$$

$$F1_{last} = score\_eval(S)$$

步骤 2：设定一个权重调整率，对于每个模型，调整的权重，根据验证集 F1 值的变化确定需要增加还是减少该模型的权重。

步骤 3：根据调整后的权重对模型进行线性融合，并重复步骤 1 和步骤 2，直到收敛。

需要注意的是，虽然只是简单的线性加权，但是由于模型的数量较多，可能会对线下的验证集过拟合，所以还需要通过线上的测试集表现来进一步确定模型融合的好坏。最后我们提交的结果中融合了 37 个模型，这些模型性能差异较大。按照最后调整的权重排序，如表 3 所示。表中的 F1 值都是线下验证集的结果（线上一般比线下高 1.5 到 2.0 个千分点）。在最后提交的模型中，线下分数为 0.43098，线上为 0.43296。

表 3. 模型融合

model_name	f1	weight
ch5-1-2embed-rnn256-cnn2345.npy	0.399	10.461
p1-1-bigru-512.npy	0.414	9.759
p1-2-bigru-512-true.npy	0.413	8.639
ch3-2-cnn-256-23457.npy	0.398	6.174
c1-1-cnn-max-256-23457.npy	0.414	5.824
ch4-1-han-bigru-256-52.npy	0.398	5.593
ch5-2-2embed-rnn512-cnn3457.npy	0.398	5.515
m9-han-bigru-title-content-512-30.npy	0.408	5.509
p5-1-2embed-rnn256-cnn2345.npy	0.406	5.467
p4-1-han-bigru-256.npy	0.408	5.300
m8-han-bigru-256-30.npy	0.408	5.165
han-cnn-title-content-256-345.npy	0.410	5.044
p3-2-cnn-256-2357.npy	0.409	4.902
p3-3-cnn-max-256-345710.npy	0.410	4.200
f1-1-cnn-256-23457-11.npy	0.412	3.723
ch6-1-han-cnn-2345-1234.npy	0.398	3.720
textcnn-fc-drop-title-content-256-3457-drop0.5.npy	0.409	2.983
m9-2-han-bigru-title-content-512-30.npy	0.403	2.842
m7-2-rnn-cnn-128-100.npy	0.404	2.760
textcnn-fc-title-content-256-345.npy	0.397	2.582
ch3-1-cnn-256-2345.npy	0.397	2.433
textcnn-title-256-len50.npy	0.390	2.106
ch7-1-2embed-rnn256-hcnn-2345-1234.npy	0.394	1.742
c1-2-cnn-256-345710.npy	0.411	1.278
p3-cnn-512-23457.npy	0.406	1.175
han-cnn-title-content-256-23457-1234.npy	0.410	0.066
p2-1-rnn-cnn-256-256.npy	0.410	-0.217
textcnn-fc-drop-title-content-256-3457-drop0.2.npy	0.405	-0.254
attention-bigru-title-content-256.npy	0.397	-0.471
c2-1-bigru-256.npy	0.405	-0.529
m7-rnn-cnn-256-100.npy	0.406	-0.794
han-bigru-title-content-256-30.npy	0.408	-0.930
m1-2-fasttext-topicinfo.npy	0.400	-1.246
textcnn-fc-drop-title-content-256-345.npy	0.408	-1.461
textcnn-fc-drop-title-content-256-345-cross3cross0.npy	0.402	-1.991
textcnn-fc-drop-title-content-256-345-cross3cross2.npy	0.402	-2.070
textcnn-fc-drop-title-content-256-345-cross3cross1.npy	0.402	-3.094

6 运行环境

主要依赖的运行环境信息如表 4 所示。

表 4. 运行环境

环境/库	版本
Ubuntu	14.04.5 LTS
python	2.7.12
jupyter notebook	4.2.3
tensorflow-gpu	1.2.1
numpy	1.12.1
pandas	0.19.2
matplotlib	2.0.0
word2vec	0.9.1
tqdm	4.11.2

7 主要文件结构

```
|- zhihu-ye-6
|       |- raw_data                 # 解压所有文件至此
|       |- data                     # 预处理得到的数据
|       |- models-notebook          # 模型代码
|       |- data_process             # 数据处理代码
|       |- ckpt                     # 训练好的模型
|       |- summary                  # tensorboard 数据
|       |- scores                   # 测试集预测概率矩阵
|       |- local_scores             # 验证集预测概率矩阵
|       |- result                   # 测试集提交结果
|       |- local_ensemble.ipynb     # 验证集模型融合
|       |- ensemble.py              # 测试集模型融合
|       |- data_helpers.py          # 数据处理函数
|       |- evaluator.py             # 评价函数
```