机器学习工程师纳米学位项目

Quora 句子相似度匹配

陈云

2019 年 11 月 06 日

目录

Ⅰ 定义

1.1 项目概述

Quora 于 2017 年在 kaggle 上公开了一个句子匹配数据集，同时基于此数据集发起了句子相似度匹配大赛，目标是为两个句子进行一致性标签标注，从而来判断句子是否一致。

句子相似度匹配是 NLP（自然语言处理）领域里的基本任务，是检索任务、对话任务、分类任务的基础。以 Quora 为例，它自身是一个知名的在线问答网站，每天有上亿的人次在 Quora 上提问，在所有这些问题当中，有相当一部分是重复的，例如：“如何进行网上购物？”和“网上购物的步骤有哪些？”这两个问题实质上是同一个问题，因为它们的目的都是一样的。这时候应该让用户看到类似的问题的回答，如果满意的话就不用再创建新的问题。

这个问题困难的地方就在于，每个人对于同样的意图会有各种各样的表达，所以文本相似度的匹配就成为了一个很有挑战性的任务。

1.2 问题陈述

该项目本质上是一个二分类问题，对于给定的句子对（question1， question2），我们需要打上“相似”或“不相似”的标签。训练数据集已经为我们做好标注。

本项目中主要的几个部分是：

- 特征工程：原始的文本并不能让模型直接处理，我们需要从句子中挖掘特征，例如基于 word2ve(词向量)[1]的特征；文本本身的特征，如句长，两个句子相同词的数量等等。
- 使用挖掘的特征在 LightGBM 机器学习算法上进行单模型实验和调节参数。
- 搭建深度神经网络，采用深度学习模型来为问句对分类。
- 进行模型融合实验，来获得更好的结果。

1.3 评价指标

评价主要采用对数损失（log-loss），计算方法如下：

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

这个指标相对于准确率（accuracy）等评价指标而言，提供了对模型更精确的评价，换句话说，模型不仅要分类分得准确，而且必须对自己的分类有足够的把握（以概率的方式表达对于预测的肯定程度）。如果对数损失低，那意味着模型分类准确而且对于分类足够地确信。

Ⅱ 分析

2.1 数据探索及可视化

- 我们的数据集数据格式如下：

Quora 训练集

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

图 2.1.1

Quora 测试集

	test_id	question1	question2
0	0	How does the Surface Pro himself 4 compare wit...	Why did Microsoft choose core m3 and not core ...
1	1	Should I have a hair transplant at age 24? How...	How much cost does hair transplant require?
2	2	What but is the best way to send money from Ch...	What you send money to China?
3	3	Which food not emulsifiers?	What foods fibre?
4	4	How "aberystwyth" start reading?	How their can I start reading?

图 2.1.2

- 数据集的总体情况如下：

	训练集	测试集
问题对总数	404290	2345796
q1 唯一问句数量	290457	2211009
q2 唯一问句数量	299175	2227400

表 2.1.1

从上表看出，我们的数据集存在着不少重复的问句，这意味着同一个问句会跟多个句子配对，由于这种关联性的存在，也许会成为判别相似性的一类特征。这一点在下面的特征工程中会提到。

- 问句的简要统计信息如下：

Quora 训练集

	Q1(char)	Q2(char)	Q1(word)	Q2(word)
count	404290	404290	404290	404290
mean	59	60	10	11
std	29	33	5	6
min	1	1	1	1
25%	39	39	7	7
50%	52	51	10	10
75%	72	72	13	13
max	623	1169	125	137

表 2.1.2

Quora 测试集

	Q1(char)	Q2(char)	Q1(word)	Q2(word)
count	2345796	2345796	2345796	2345796
mean	60	60	11	11

std	31	31	5	5
min	1	1	1	1
25%	40	39	7	7
50%	53	52	10	10
75%	72	72	13	13
max	1172	1176	238	238

表 2.1.3

注：Q1(char)指的是 question1 的字符长度统计，Q1(word)指的是 question1 的单词（按空格切分）个数统计。

从上面两张表可以看出，大部分的句子都是短句子，词的个数不超过 20 个。测试集与训练集的统计特性类似，说明从训练集来预测测试集是可行的。

- 在训练集中的正负样本分布如下：

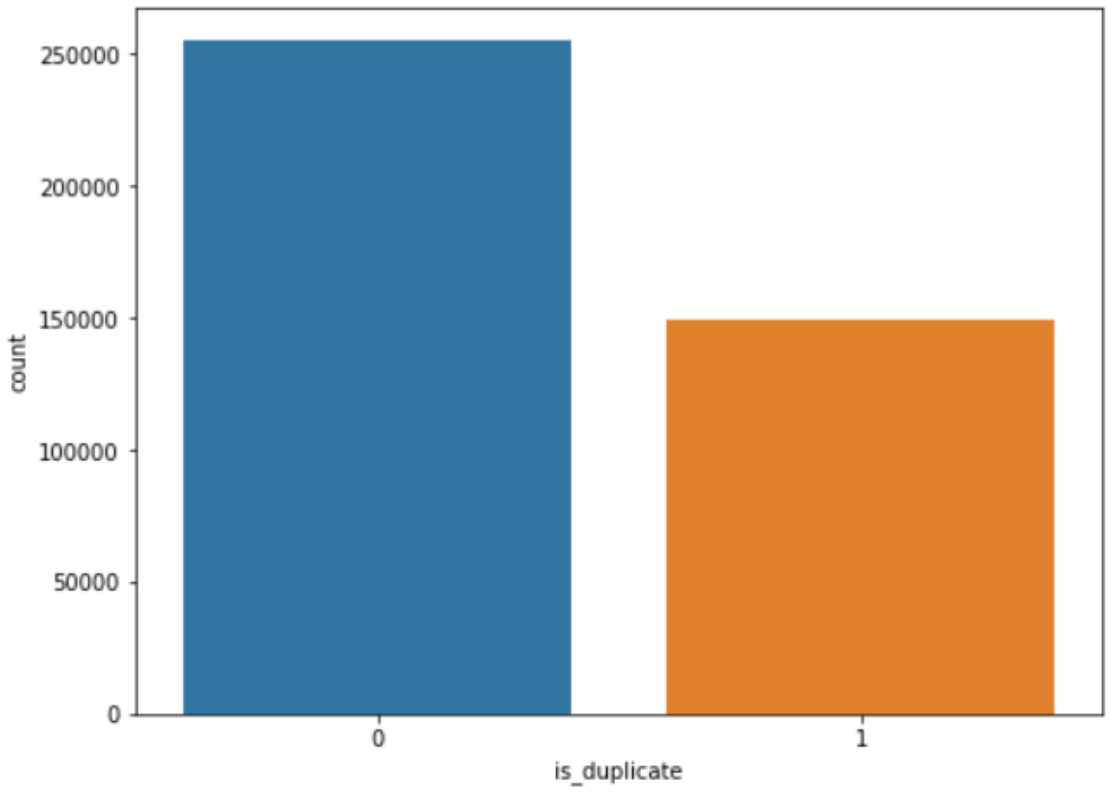


图 2.1.3

2.2 算法和技术

创办于 2010 年的 Kaggle 平台是世界上最大的机器学习竞赛平台之一，本项目就是 2017 年 Quora 在 kaggle 上发起的一场比赛。Kaggle 上的竞争非常激烈，而且涵盖了各种类型的机器学习问题，所以通过观察 Kaggle 竞赛来了解机器学习算法的发展情况也是很不错的方式。

在 2016 和 2017 年，Kaggle 上主要有两大方法：梯度提升和深度学习[2]。引用 keras 之父——弗朗索瓦·肖莱的话，要想在如今的应用机器学习中取得成功，你应该熟悉两种技术：梯度提升机，用于浅层学习；深度学习，用于感知问题。

在本项目中，我采用了基于梯度提升的 LightGBM 算法和深度学习技术。

2.2.1 LightGBM 算法

LightGBM 是微软团队于 2017 年发表在 NIPS(人工智能方面的顶级会议，现更名为 NeurIPS)上的一篇论文。它在保证模型精度的情况下，大大提高了 GBDT（梯度提升决策树）类模型的算法效率，从而引发了学界和工业界的广泛关注。由于篇幅所限，这里只介绍 LightGBM 的两个特点，更详细的理论请参看论文原文[3]。

- **直方图算法(Histogram)**

直方图算法的基本思想是先把连续的特征值离散化成 k 个整数，同时构造一个宽度为 k 的直方图。在遍历数据的时候，根据离散化后的值作为索引在直方图中累积统计量，当遍历一次数据后，直方图累积了需要的统计量，然后根据直方图的离散值，遍历寻找最优的分割点。

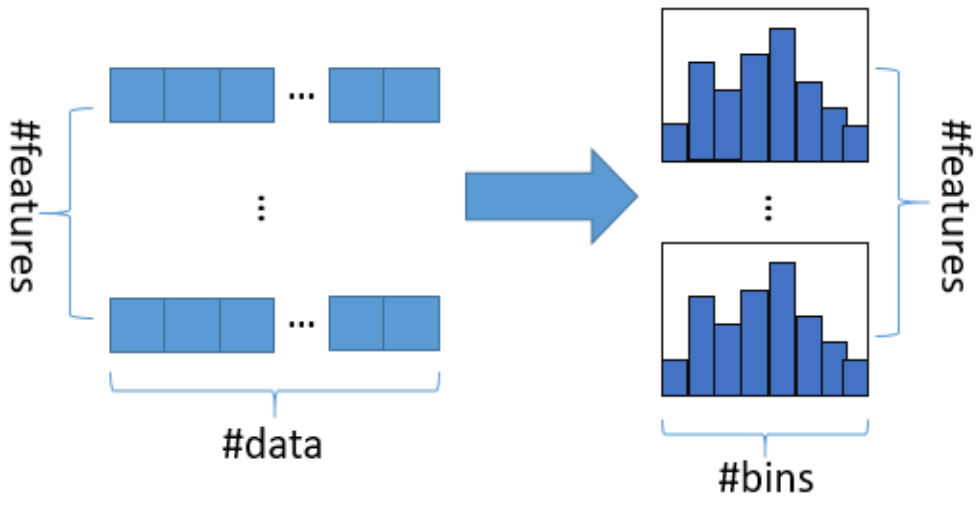


图 2.2.1.1

虽然特征被离散化后，分割点不是很精确，但是 Boosting 框架本身就是一系列弱模型的集成，弱模型上精度的损失反而带来了一些正则化的效果，防止过拟合。

相对于 Xgboost（GBDT 的最好实现之一）使用的 pre-sorted（预排序）算法[4]，直方图算法将分割增益的计算量从 O(#data)降低到了 O(#bins),#bins 远远小于#data。

还可以通过直方图相减来进一步加速训练，在二叉树中可以通过利用叶节点的父节点和相邻节点的直方图的相减来获得该叶节点的直方图：

图 2.2.1.2

- Leaf-wise 决策树生长策略

大部分决策树的学习算法通过 level-wise 策略生长树，也就是一次分裂同一层的叶子，不加区分的对待同一层的叶子，而实际上很多叶子的分裂增益较低没必要进行分裂，带来了没必要的开销：

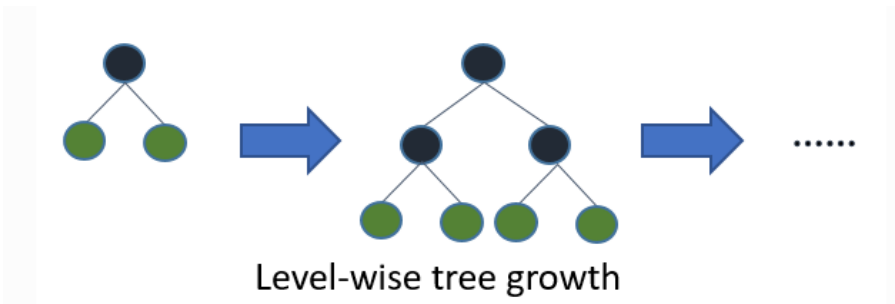


图 2.2.1.3

LightGBM 通过 leaf-wise 策略来生长树。每次从当前所有叶子中，找到分裂增益最大的一个叶子，然后分裂，如此循环。因此同 Level-wise 相比，在分裂次数相同的情况下，Leaf-wise 可以降低更多的误差，得到更好的精度：

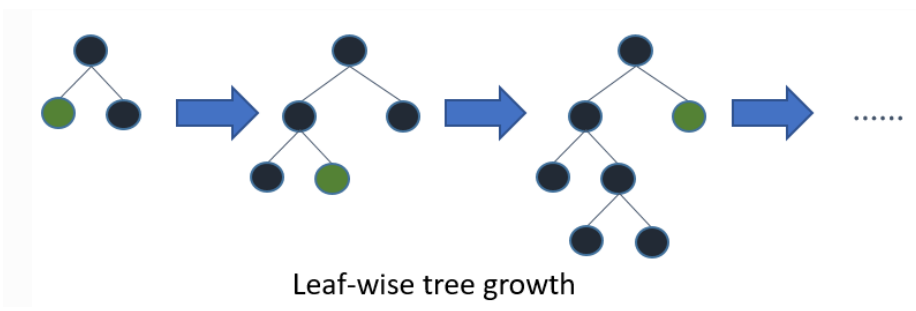


图 2.2.1.4

2.2.2 深度学习技术

深度学习是机器学习的一个分支领域：它是从数据中学习表示的新方法，强调从连续的层（layer）中进行学习，而这些层对应着越来越有意义的表示。数据模型中包含多少层，被称为模型的“深度”（depth）。现代的深度学习方法往往包含着数十个甚至上百个表示层，这也是它跟其他机器学习方法区分开来的重要特征。

深度学习的一个很大的优势在于，我们不必再去做复杂的特征工程，可以让模型自动从数据中找到特征，这大大增强了机器学习在实践中的应用能力，只要给深度模型提供足够多的数据，往往就能得到神奇的效果。

2.2.2.1 长短期记忆网络（LSTM）

LSTM 是 RNN(循环神经网络)的一种改良。所以在讲 LSTM 之前我们先看看 RNN 的结构：

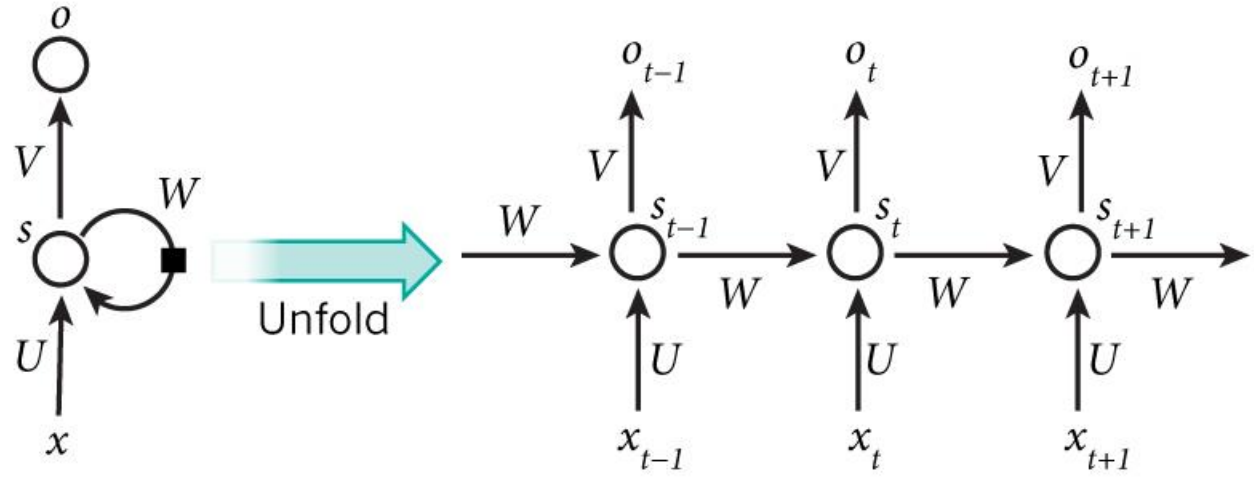
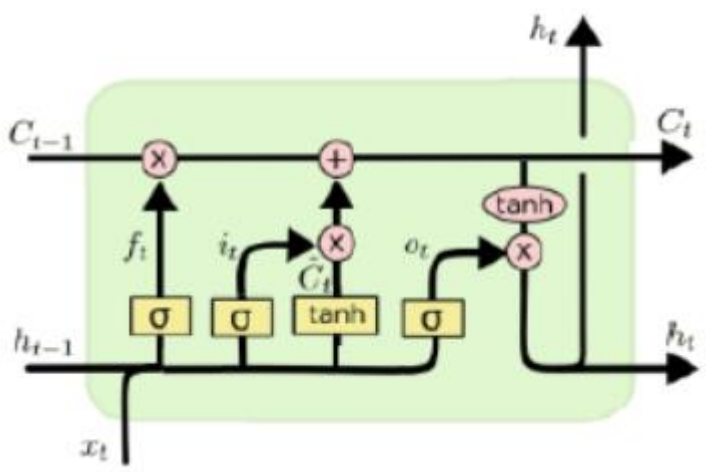


图 2.2.2.1.1

RNN 与其他网络的不同之处在于，它是循环结构，它把前一次迭代的计算结果考虑了进去，相当于赋予了模型记忆功能：

- X(t)是时间 t 处的输入
- S(t)是时间 t 处的“记忆”，S(t) = f(U • X(t) + W • S(t-1))，这里的 f 是激活函数，常用 tanh
- O(t)是时间 t 处的输出，例如要预测下一个词是什么的话，可以是 softmax 输出的每个候选词的概率，O(t) = softmax(V • S(t))

理论上来说，RNN 应该能够记住许多时间步之前的信息，但在实践中并不是，其原因在于梯度消失问题（vanishing grandient problem）[5]。LSTM 就是为了解决这个问题产生的设计。



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

图 2.2.2.1.2

- C(t)代表了 LSTM 内部的“记忆”
- f(t):Forget Gate,遗忘门。表示是否保存当前隐层节点的历史信息，1 则允许输入，0 则清空信息
- i(t):Input Gate,输入门。表示是否允许信息加入到当前隐层节点中
- o(t):Output Gate,输出门。表示是否将当前节点的输出值输出给下一层
- 通过这些设置，即保存了更长期的记忆，也能“遗忘掉”不重要的记忆

我们可以把上图中的 C 想象它处于一个传送带上，序列中的信息可以在任意位置跳上传送带，然后被传送到更晚的时间步，并在需要时原封不动地跳回来。总结来说就是，LSTM 保存信息以便后面使用，从而防止较早期的信号在处理过程中逐渐消失。

2.2.2.2 双向 LSTM

在单向的循环神经网络中，模型从时间步是顺序流逝的，也就是从之前的时间步进行学习。而在某些任务中，例如自然语言理解，我们需要从上下文中理解句子的完整意思。比如有这么两句话:“他说苹果很好吃”和“他说苹果手机很好用”，如果仅仅看到苹果的话，我们还不知道指的是水果还是手机，这时候我们就需要从未来的时间步中进行学习，以理解上下文信息。

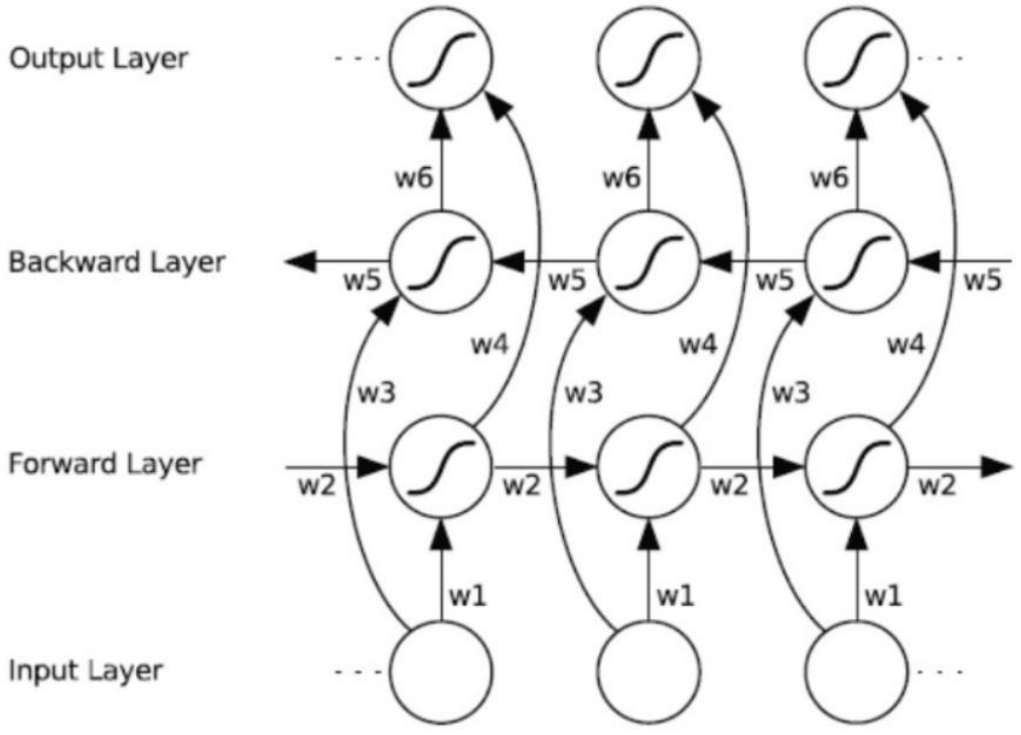


图 2.2.2.2.1

2.2.2.3 孪生网络（Siamese Network）

孪生网络很适合用来解决相似性的问题，它的网络结构如下：

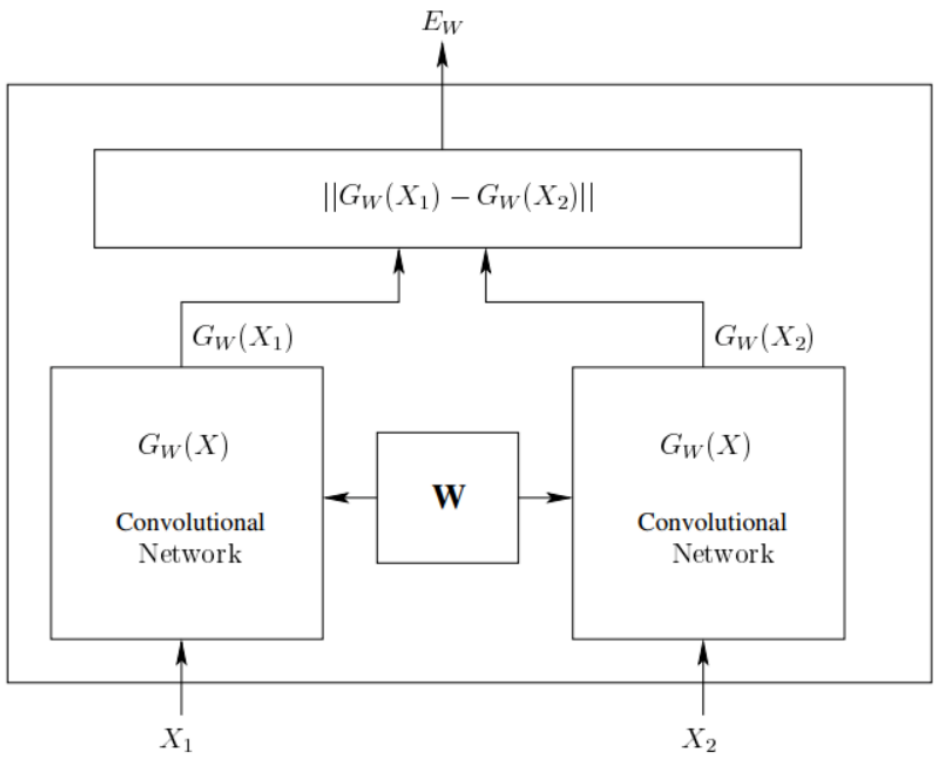


图 2.2.2.3.1

它有两个输入，每个输入都经过了相同的网络结构（图中的是卷积神经网络，也可以是其他网络，如 LSTM），得到了两个新的表示，然后对两个新的表示使用某种相似性度量。两个网络的权值是共享的（图中的 W），这也减少了模型需要训练的参数量。

2.2.2.4 一维卷积网络（1D-CNN）

近年来，卷积神经网络（CNN）因在视觉图像领域取得的巨大成功而为人们所熟知。在视觉图像领域一般用的是二维卷积（这里不对卷积网络多做展开，原理可看这篇综述性论文[6]），而一维卷积层可以用在序列模型中，特别适合自然语言处理的领域。

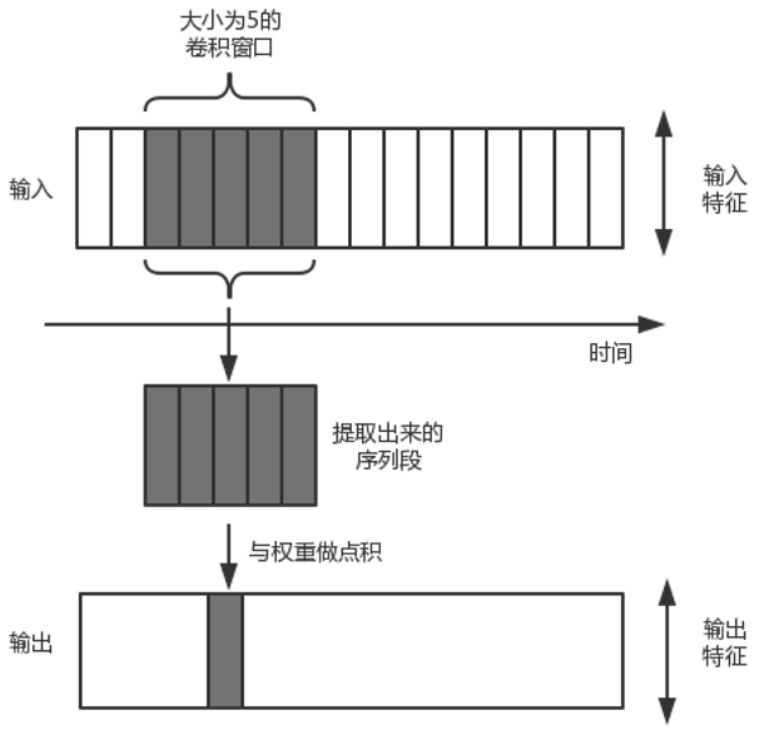


图 2.2.2.4.1

一维卷积可以识别序列中的局部模式。因为对每个序列段执行相同的输入变换（权值共享），所以在句子中某个位置学到的模式，稍后可以在其他位置被识别，这使得一维卷积神经网络具有平移不变性（相对于时间平移而言）。

如上图所示，使用大小为 5 的卷积窗口处理字符序列的一维卷积网络，应当能够学习长度不大于 5 的单词或者单词片段，并且应该能够在输入句子中的任何位置识别这些单词或者片段。因此，字符级的一维卷积神经网络能够学习单词构词法。

2.3 基准指标

该项目在 kaggle 上有 3307 组参赛者进行比拼，最终成绩以 private leaderboard 上公布的为准，评价方式是 logloss。本项目的目标是进入 private leaderboard 上的 top 20% ，也就是前 660 位，对应的 logloss 得分是 0.18267。

III 具体方法

3.1 特征工程

3.1.1 文本特征

主要是句子本身的一些特征，列表如下：

feature_name	describe
q(1,2)_len	问句 1 或 2 的总字符长度
q(1,2)_char_len	问句 1 或 2 的有效字符长度（去掉空格）
q(1,2)_word_len	问句 1 或 2 的单词个数（空格切分）
char_len_diff	两问句字符长度差
word_len_diff	两问句单词个数差
common_words	两问句的共同单词个数

表 3.1.1.1

3.1.2 编辑距离（Edit distance）特征

所谓的编辑距离算法是指：两个字符串之间，由一个转成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。一般来说，编辑距离越小，两个串的

相似度越大。

我使用了 fuzzywuzzy 这个 python 库来计算句子的编辑距离。

3.1.3 词向量（word2vec）特征

在词向量之前，传统 NLP 对于词的表示方法是高维稀疏的 one-hot 表示方法，这种方法将单词散列编码为固定长度的向量，这个单词所在的位置为 1，其他全为 0，如果词典很大，那这是一个非常稀疏的向量，而且单词与单词之间是独立的，没有什么顺序上的关联。

而词向量的提出很好地解决了这个问题[1],通过一个简单的神经网络结构，将 one-hot 编码形式的高维稀疏向量映射成了浮点型的低维稠密向量，而且这样的映射也考虑到了上下文关系，词与词之间不再是割裂的。对于生成词向量的神经网络，其实我们并不关心最后的输出结果，网络训练好之后，中间层的权重矩阵才是我们需要的东西——词向量。

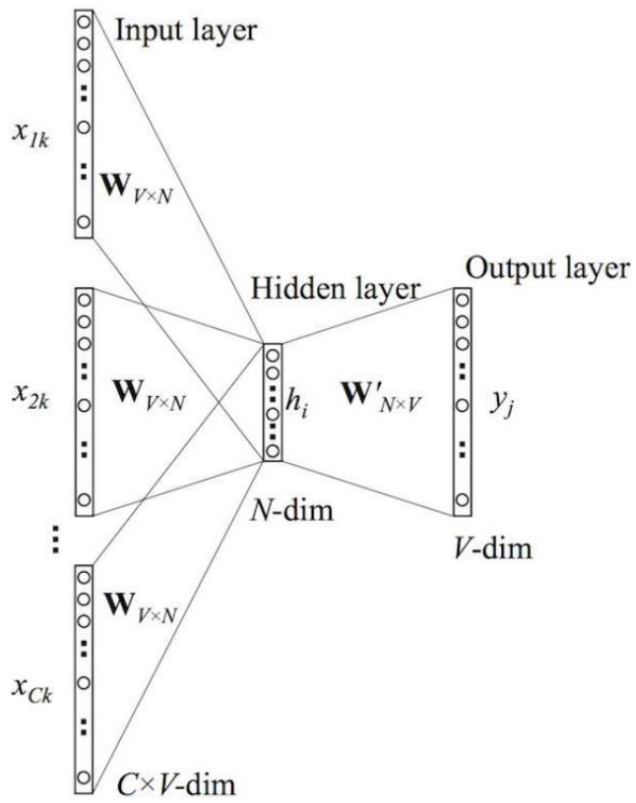


图 3.1.3.1

当我们获得词向量之后，就可以基于它来进行词和句子的比较，例如计算词移距离（Word Mover's Distance），如下图所示，去掉停用词后，文档 1 基于词向量转换到文档 2 所花费的代价就是它们之间的词移距离。

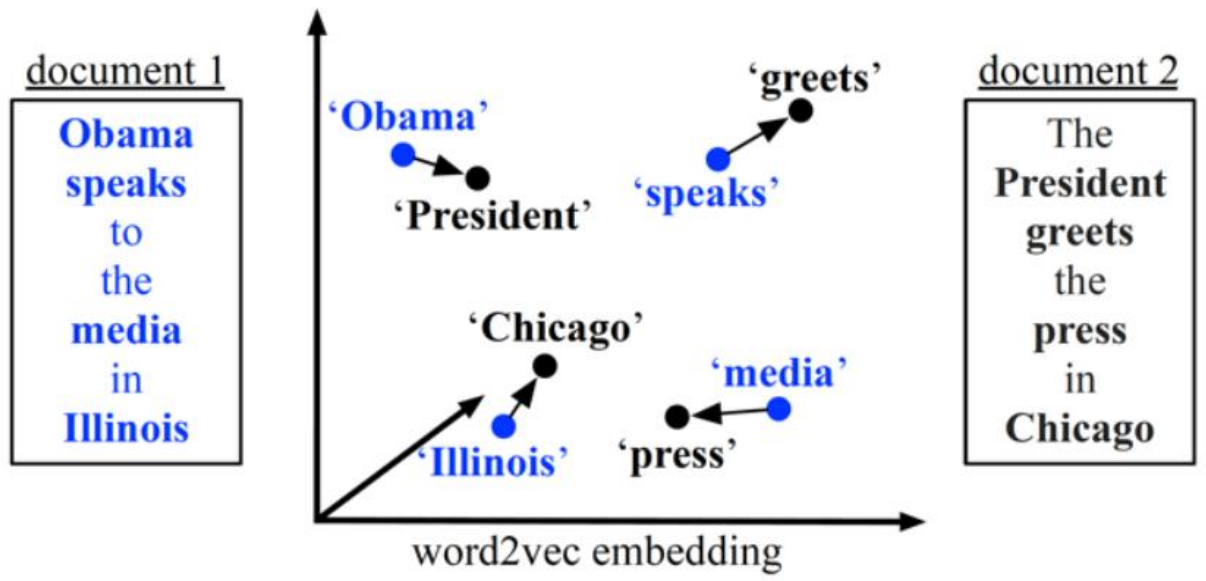


图 3.1.3.2

3.1.4 图特征

这个部分的特征其实跟自然语言处理本身关系不大，只是数据集本身体现了某种规律。在前面的数据探索部分，我们发现有大量重复的问句，那这些句子之间可能存在着某种网络属性。有参赛者根据这个思路将问句建立成了图结构（句子作为节点，句子对作为边），去挖掘这个图的特征，例如句子相互连接会聚集成 clique（团），这个团的大小就可以作为句子的特征。

3.2 LightGBM 模型

本项目采用了 python 版的 lightGBM（lightgbm-2.2.3）。具体的步骤如下：

- 将特征工程中生成的特征组合起来，区分训练集（X,y）和测试集（X_test）
- 使用 sklearn 中的 train_test_split 模块将训练集拆分为 80%的训练数据(X_train,y_train),20%的验证数据(X_valid,y_valid)
- 将训练数据和验证数据都转换到 lightgbm 的专有数据结构 Dataset 中：

```
lgbm_train = lgbm.Dataset(X_train, y_train)
lgbm_valid = lgbm.Dataset(X_valid, y_valid, reference=lgbm_train)
```

- 使用 train 函数训练数据，使用 predict 函数预测测试集

LightGBM 中调节的超参数（部分）如下（由于时间关系只尝试了部分）：

parameter	describe
num_leaves	树的最大叶子节点数量
learning_rate	学习速率
bagging_fraction	构建决策树时，样本采样的比例
bagging_freq	进行 bagging 的频率
min_data_in_leaf	叶子节点可能具有的最小记录数
max_bin	表示 feature 将存入的 bin 的最大数量

表 3.2.1

3.3 深度学习网络模型

我为本项目设计的深度学习网络结合了双向 LSTM、孪生网络，以及 1D-CNN，希望通过引入不同的表达来使得模型的结果更为准确。具体的网络结构如下：

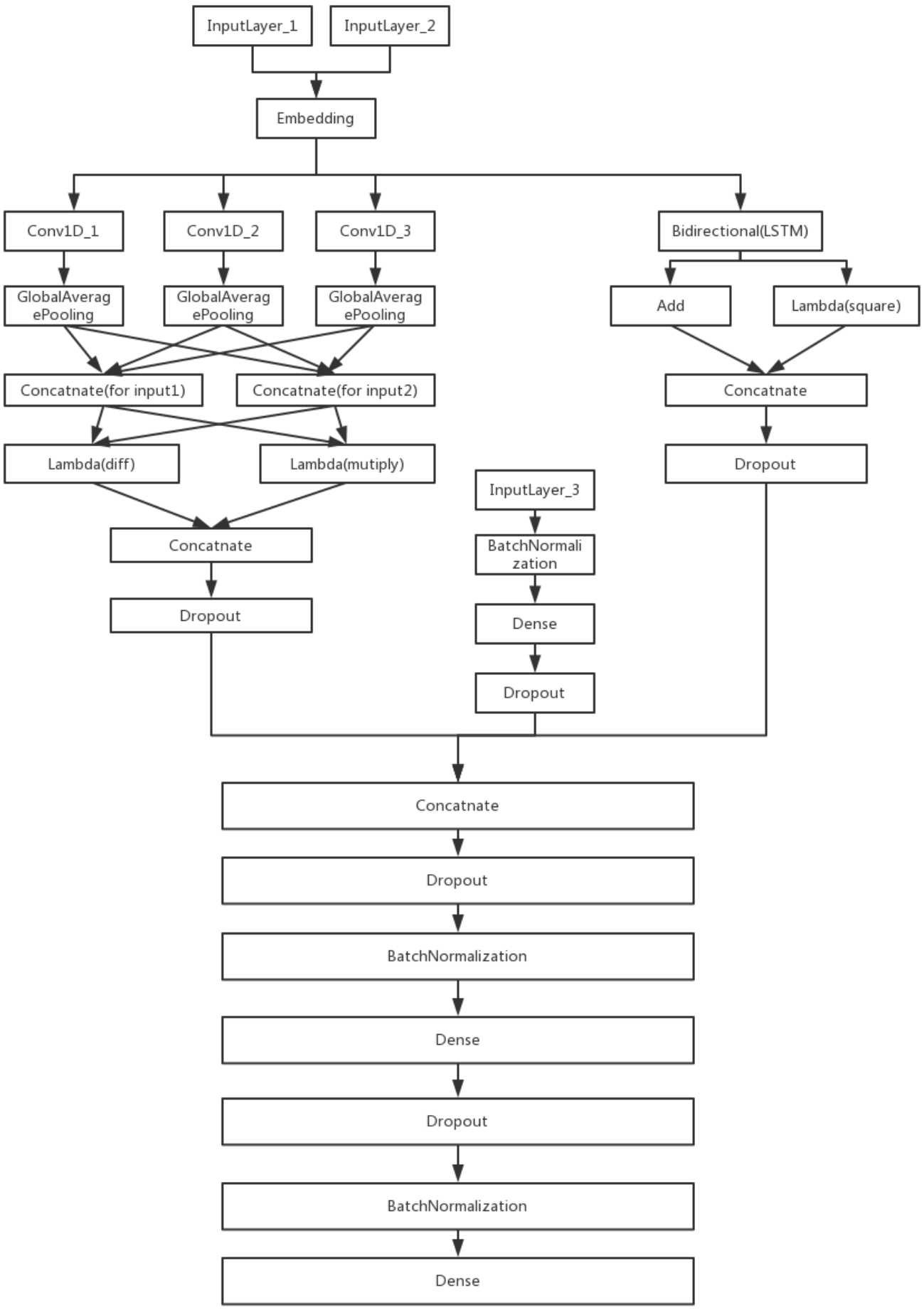


图 3.3.1

图中的 Input_1 和 Input_2 分别对应了数据集中的 question1 和 question2。当然，我们并不能直接将句子送入模型中，而是先把它们转换为一个固定长度的标记序列，例如：

[i,love,deep,learning] —> [0,0,0,0...4,23,12,56]

右边的数字是左边的词在词典中的编号，由于是固定长度，不足的部分补 0。

网络主要分为三个部分，1D-CNN 和双向 LSTM 部分都采用了孪生网络的方式，也就是句子对中的两个句子都会通过相同的网络，权值共享：

- 1D-CNN

该部分采用了三种不同卷积窗口的 1 维卷积网络，相当于在感受野中一次观察不同长度的词序列：

```
conv1 = Conv1D(filters=64, kernel_size=1, padding='same', activation='relu')
conv2 = Conv1D(filters=64, kernel_size=3, padding='same', activation='relu')
conv3 = Conv1D(filters=64, kernel_size=5, padding='same', activation='relu')
```

接着，每一个卷积网络都接了一个全局平均池化层进行降维（3 维->2 维），相当于对**每一个输入**（句子），都获得了 3 个 64 维的向量，**拼接**后(concatenate)是一个 192 维的向量，可以理解为模型通过这几个网络帮我们“总结”出了 192 维的特征。因为是要比较两个句子的相似度，所以我采用了**两个** 192 维向量的算术合并来进行衡量：

Lambda(lambda x: K.abs(x[0] - x[1]))([concat_1, concat_2]) —> 逐元素差的绝对值

Lambda(lambda x: x[0] * x[1])([concat_1, concat_2]) —> 逐元素积

- Bidirectional(LSTM)

双向 LSTM 的设置如下：

Bidirectional(LSTM(64))

LSTM 层的输出设置为 64 维，采用了双向包装层后，输出变成了 128 维，相当于我们的句子经过双向 LSTM 结构后，生成了 128 维的特征向量。由于 LSTM 和 1D-CNN 的内在原理不一样，所以它们从输入中学习到的东西是不一样的（虽然结果可能相似），是对待同一个输入的两种不同的观点。同样，这里我也采用了**两个** 128 维向量的算术合并，不过为了增加多样性，使用了和 1D-CNN 结构不一样的合并：

Lambda(lambda x: K.square(x[0] - x[1]))([lstm_1, lstm_2]) —> 逐元素差的平方

add([lstm_1, lstm_2]) —> 逐元素和

- **特征部分**

这一部分比较简单，就是把在特征工程部分做好的特征向量输入到网络中。这样做的目的同样是为了增加不同的观点，希望能增强模型的整体能力

3.4 改进

在前面的实验中，只是进行了单模型的训练，为了进一步提高效果，采用模型融合的方式来进一步提高效果。具体的方式如下：

- 将训练集拆分为 10 份，生成 10 个模型，每个模型使用其中的 9 份数据作为训练数据，剩下一份作为验证数据。
- 训练好的 10 个模型分别对测试集进行分类，获得了 10 份分类结果。
- 将 10 份分类结果加总平均，获得最终的预测结果。

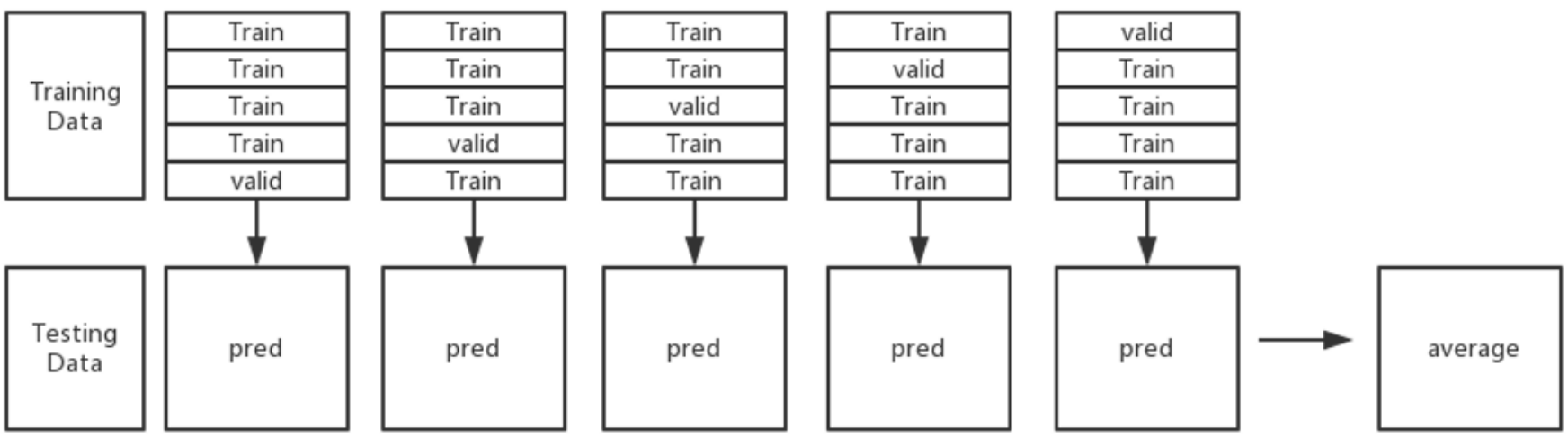


图 3.4.1

Ⅳ 结果

4.1 模型的评价与验证

LightGBM 模型的 ROC 曲线和 AUC 值：

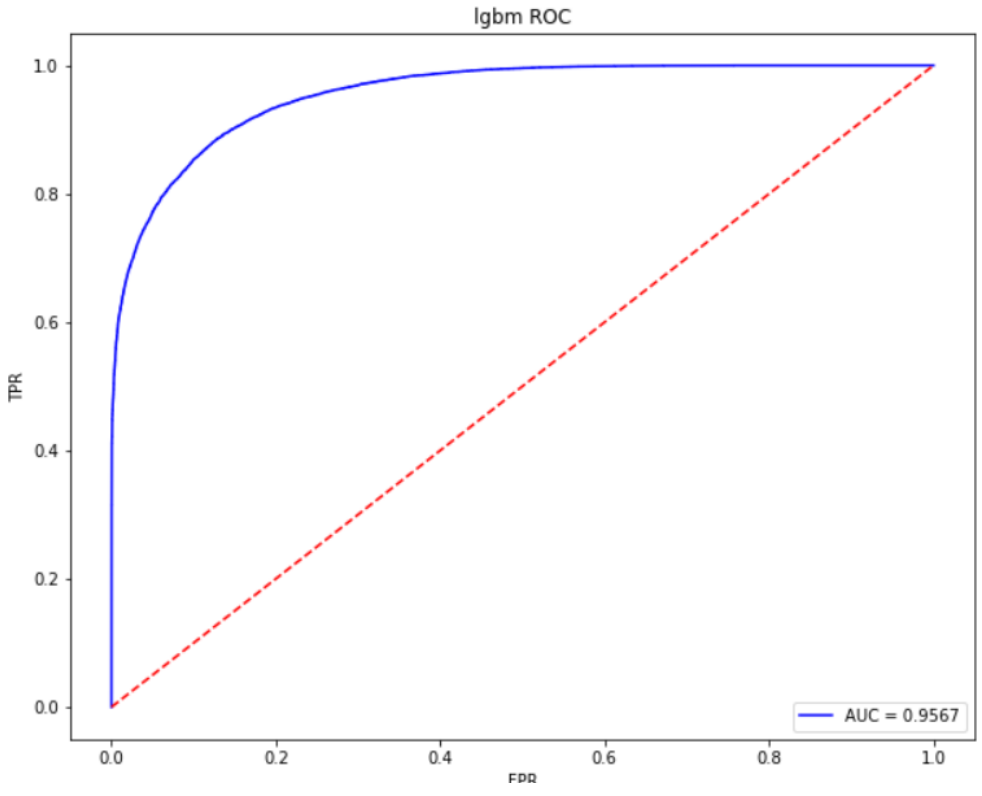


图 4.1.1

深度神经网络的 ROC 曲线和 AUC 值：

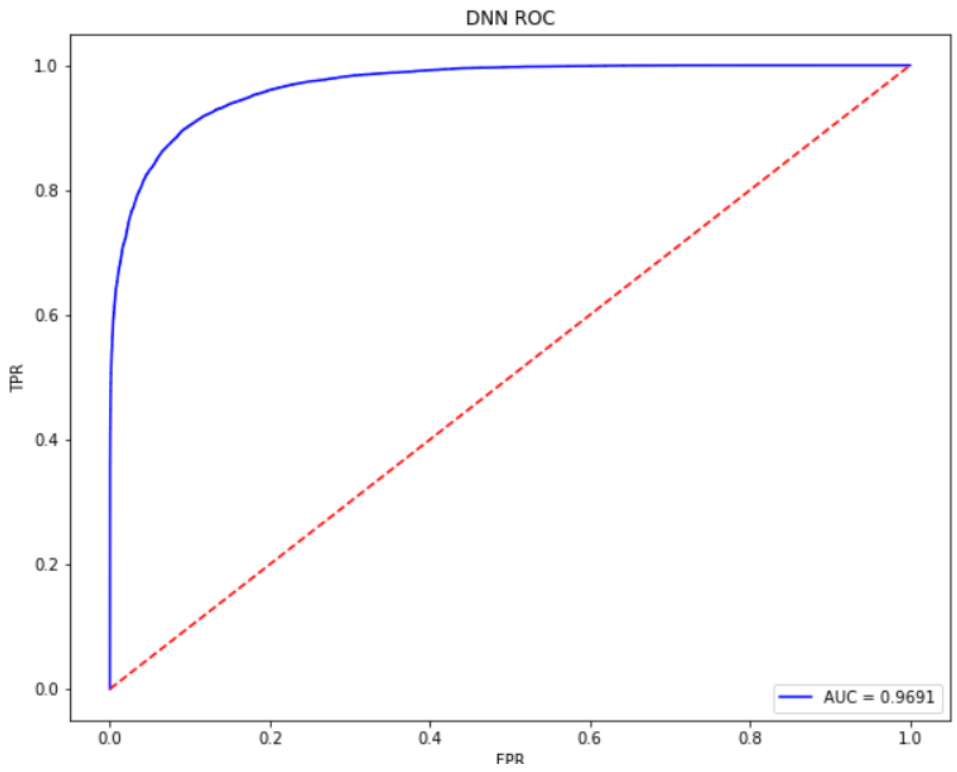


图 4.1.2

在 ROC 曲线中，横轴 **FPR**（false positive rate） 越大，预测正类中实际负类越多；纵轴 **TPR**（true positive rate）越大，预测正类中实际正类越多。

我们的理想目标是 FPR=0，TPR=1,也就是图中的（0，1）点。所以 ROC 曲线越靠拢这个点，越偏离 45°对角线越好。而这也意味着 ROC 曲线下的面积越大越好，这就是 AUC 值。

从图上来看，两种模型的效果都还不错，很靠近（0，1）点，深度神经网络的 AUC 值略高于 LightGBM 模型，说明其分类性能更佳。

4.2 结果分析

在本项目中，我们实验了 LightGBM 单模型，DNN 单模型，以及两个模型分别的融合模型。以下是各个模型在 kaggle 上提交的分数：

	Public Leaderboard	Private Leaderboard
LightGBM 单模型	0.20120	0.20489
DNN 单模型	0.15276	0.15564
LightGBM 模型融合	0.16195	0.16536
DNN 模型融合	**0.14781**	**0.14936**

表 4.2.1

最优提交的截图：

图 4.2.1

从提交的结果来看，深度神经网络的单模型表现比模型融合后的 LightGBM 模型还要更好，而模型融合后的 DNN 在此次实验中表现最佳，这个成绩进入了 Private Leaderboard 的前 200 名，属于 top 6%，达到了我们的预期目标。

Ⅴ 结论

深度类模型在本项目中展现了强大的能力，查看 kaggle 上排名靠前的解决方案，基本都使用了深度神经网络。但这也不意味着特征工程不再重要，好的特征工程再配合深度学习模型，可以让模型的性能有很大提升。

在本项目的实践中还有许多可以改进的地方：

- LightGBM 提供了 sklearn 的接口，可以尝试使用 GridSearchCV 来进行自动调参
- 句子的拼写纠错，符号处理等
- 提取更多的特征
- 设计不同类型的网络结构进行实验
- 在最后的模型融合部分只是使用了简单的求和平均，可以尝试更高级的融合方式，例如 Stacking，将多个学习器堆叠起来，预测的结果作为特征让下一层的学习器进行学习

Ⅵ 参考文献

[1]. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representation in Vector Space, 2013

[2]. François Chollet:《Deep Learning with Python》

[3]. A Highly Efficient Gradient Boosting Decision Tree

[4]. Mehta, Manish, Rakesh Agrawal, and Jorma Rissanen. “SLIQ: A fast scalable classifier for data mining.” International Conference on Extending Database Technology. Springer Berlin Heidelberg, 1996.

[5]. Bengio Y, Simard P, Frasconi P: Learning long-term dependencies with grandient descent is difficult [C]//IEEE 1994

[6]. Recent Advances in Convolutional Neural Networks