# “句子相似度匹配”项目报告

凌塑奇 2019.08.23

## 问题的定义

### 项目概述

“句子相似度匹配”项目属于自然语言处理（NLP）领域。顾名思义，该领域就是利用计算机处理人类语言（自然语言）。所谓“处理”则包括分析、理解自然语言文本，以及理解自然语言文本后生成自然语言文本。本项目仅涉及前者。

自然语言处理被誉为人工智能的明珠。语言作为人类最重要的工具之一，是破解“智能”的关键。然而，纳米学位中相关内容并不多。开展这一项目将有助于了解该领域的研究动态。。

本项目需要解决的问题是判断一对句子的意思是不是相同的。这些句子对来自Quora 2017年在kaggle举办的Quora问题匹配比赛。参赛者需要给出一系列句子对是相同的意思的概率。解决该问题，将帮助Quora归并同样问题和构建更好的相关问题推荐系统。

### 问题陈述

该问题可以看作一个监督学习中的分类问题，通常的解决方案是通过特征工程将文本转变为数值特征，训练出相应的分类器。在自然语言处理领域，通常被归纳为文本语义相似性分析。

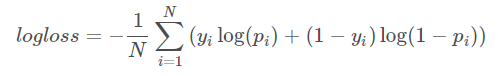
简单地说，就是把文本转换为一系列的特征（通常为数值特征），这些特征将文本的词语含义、词语之间的关系、词语的稀缺性、语法等信息隐性地用数字表示。之后，可以利用分类器通过对数字进行分析，获得句子相似的概率。基本的分类器包括线性回归、支持向量机、决策树等。

问题的解决策略将分为三步。第一步，对文本进行编码，提取隐含文本相似性的特征信息。第二步，利用不同的分类器对特征信息进行分析，挑选最合适的分类器。第三部，对最合适的分类器进行优化，建立最终的分类模型。

期望的结果是将预测的句子相似的概率提交到kaggle后，需要达到kaggle private leaderboard 的top 20%。

### 评估指标

项目使用log-loss指标作为评估指标。该指标可用如下方程描述：



(来源: <https://www.kaggle.com/c/bnp-paribas-cardif-claims-management/overview/evaluation>)

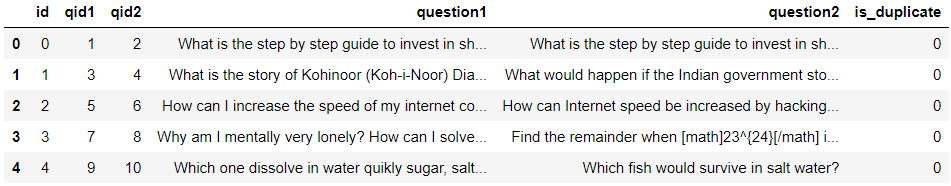
其中，N表示数据数量，yi 为i数据的真实标签，pi 为预测的i数据标签为yi 的概率。

该指标对确切的错误分类的惩罚较大。例如，预测所有数据为真实标签的概率为1的log-loss为21.79，预测所有数据为真实标签的概率为0的log-loss为12.75，而预测所有数据为真实标签的概率为0.5的log-loss为0.69）

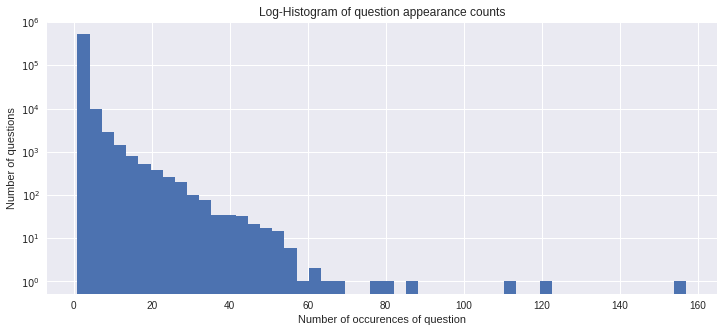
## 分析

### 数据的探索

项目的训练数据是约40万对句子对，包含句子对编号，每个句子的编号，句子文本和句子对是否相同的标签。其中，约25万对为不相同句子对、约15万对为相同句子对，相同句子对概率约为40%。前五个句子对数据如下所示：



在数据集中，大部分句子都只出现了一次，少数句子出现了很多次。句子出现次数的直方图如下所示：

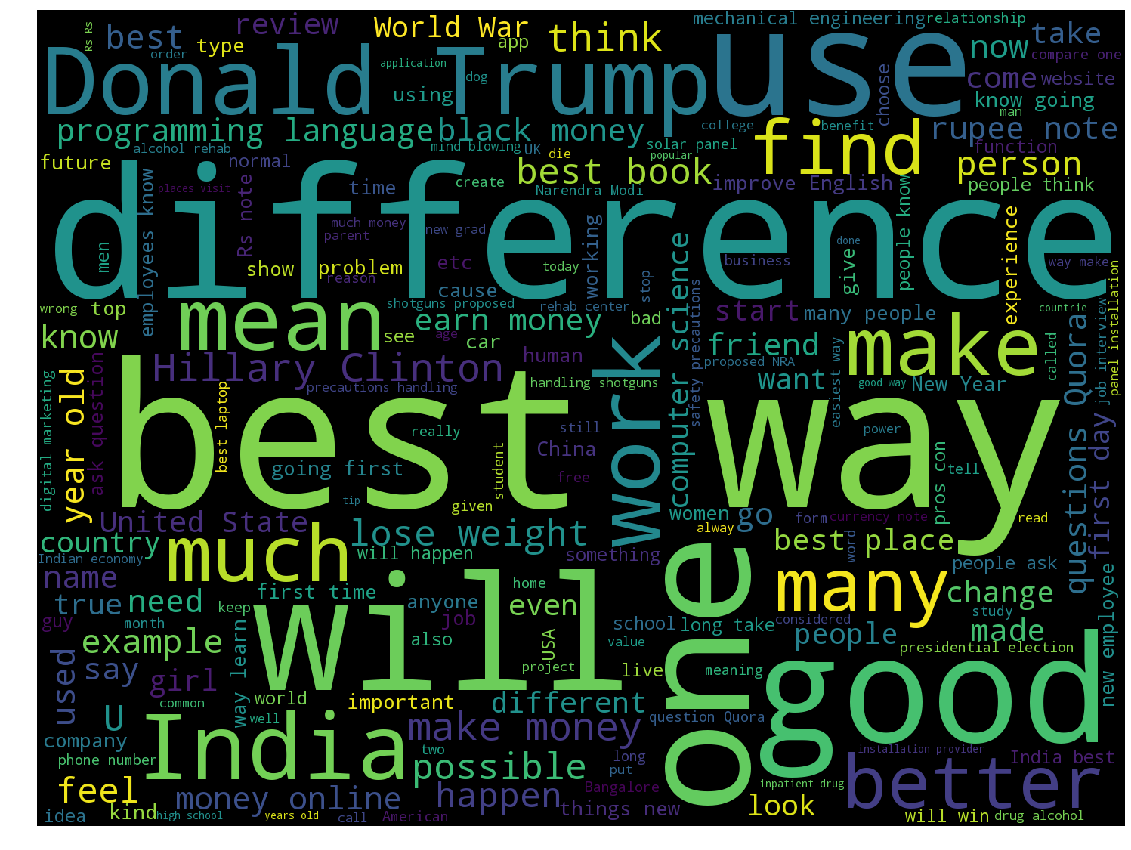


（图片来自<https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb>）

数据集中的句子平均长度约60个字符，最少字符为1个，最长字符为1169个。

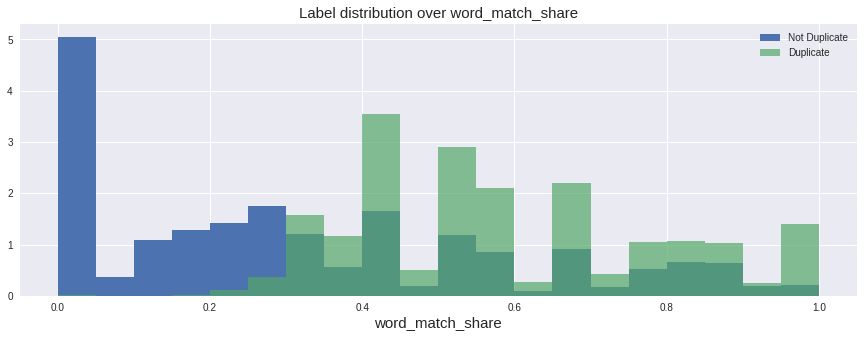
### 探索性可视化

统计数据集中的单词的出现次数绘制成如下的云图。从图中看出，出现最多的前三个单词是“difference”、“best”、“way”。



（图片来自<https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb>）

下图分别统计了被标记为不同的句子对和被标记为相同的句子对之间，包含相同单词占句子对单词数的比率。从图中可以看出，大部分被标记为相同的句子对中相同单词的比率较大。



（图片来自<https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb>）

### 算法和技术

目前常用的文本语义相似性分析有两种方法。

一种是通过特征工程对文本进行数据挖掘，将文本转换为大量的数值特征。目前的数据挖掘方法有三种。一是基于词袋模型的TF-IDF、SDV等；二是基于贝叶斯模型的LDA；三是通过深度学习将文本转换为向量的词嵌入方法，例如word2vec、GloVe等。获得文本的数值表示后，可以计算文本向量间的L1、L2范数，余弦距离，word mover distance等。

另一种是在把文本转变到向量空间的基础上，将文本向量输入深度神经网络进行训练，建立神经网络语言模型，从而实现文本相似性的分类。目前的神经网络语言模型大多Siamese 和Attention两种架构。同时，一些预训练的语言模型也在文本语义相似性上得到了良好的结果。例如Google推出的BERT在多项NLP任务上获得了state-of-art的表现。

本项目将讨论多种方法的表现，最终选用其中一种进行优化。

#### 特征工程

特征工程方面，本项目对文本进行了统计分析，TF-IDF转换、universal sentence encoding、GloVe进行向量化后计算了cityblock距离、Euclidean距离和cosine距离。

统计分析主要包括两个句子的单词数差异及相应的比率，字母数差异及对应的比率，共同单词数及对应比率，以及Jaccard相似度。

TF-IDF是term frequency – inverse document frequency的缩写。TF表示文档中某个词出现的次数，这里简单理解为词出现的次数越多，越重要。IDF表示文档总数除以所有文档中出现这个词的文档数。这两者相乘，就表示某个词在总的文档中不常见，却在某个文档中出现，那么这个词对这个文档就比较重要。

Universal sentence encoding是Google推出的文本embedding技术，该技术将句子转换为固定长度的向量。通过神经网络从大量文本中学习转换方法。选择这种embedding 方法是由于该方法不仅包含了整个句子的语义信息，而且实现非常简单。

GloVe是Stanford推出的词向量embedding模型，本项目使用了在通用文本上训练的词向量库 [glove.840B.300d.zip](http://nlp.stanford.edu/data/wordvecs/glove.840B.300d.zip) 。该库包含2.2M个单词向量，向量维度为300。相比word2vec，GloVe在一般性的NLP任务的表现更好。

#### 模型对比

完成特征工程后，主要考虑了三种模型进行对比。包括随机森林算法、XGB和预训练的BERT模型。选择这三种算法的理由是：随机森林算法是常用的分类器，XGB在近来的分类任务表现优秀，BERT在多项NLP任务获得了最佳表现。

项目中随机森林模型将采用默认参数，XGB模型采用来自Zhu Kai分享的[kaggle kernel](https://www.kaggle.com/benjaminkz/quora-question-pairs-xgboost/data)的XGB参数，BERT将文本输入预训练的BERT模型进行微调。

#### 运行环境

由于项目对的计算性能需求较高、需要GPU支持，因此选择使用云计算平台进行计算。考虑到CoLab对BERT支持较好，且提供免费TPU，因此选择在CoLab计算，数据储存在Google Cloud Storage。

Colab 的普通环境，RAM最大为25G；GPU环境RAM最大为12G；TPU环境，RAM最大为12G。

### 基准模型

BERT在多项NLP任务都有较好的表现，因此对其在本项目的表现非常好奇。本项目选择将训练数据输入BERT预训练模型进行微调作为基准模型。训练参数不做改动。

该模型在CoLab TPU 支持下训练花费1小时，在验证集的logloss为0.4958.

## 方法

### 数据预处理

首先，是对数据进行清理。包括将文本全部转换为小写，去掉特殊符号，展开常见的缩写，将常见的同义词、短语转换为统一的形式。

然后，将文本分别进行stemming、lemmatizing、去除stop words处理，并分别保存经过处理的文本。在处理过程发现，有些句子是空的，pandas读取时会转换为NaN，在处理时会报错。解决方法是使用空字符来填充NaN的数据。

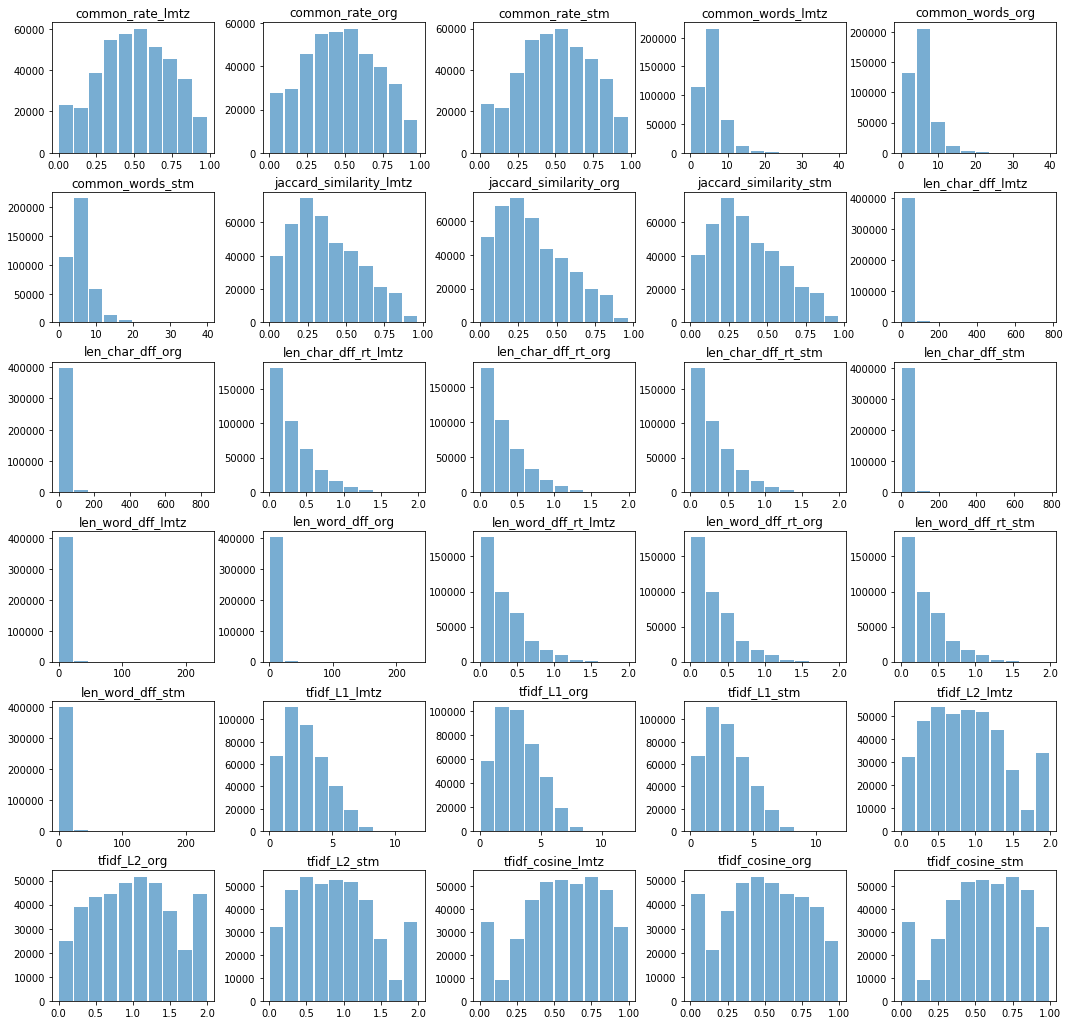
### 执行过程

#### 特征工程

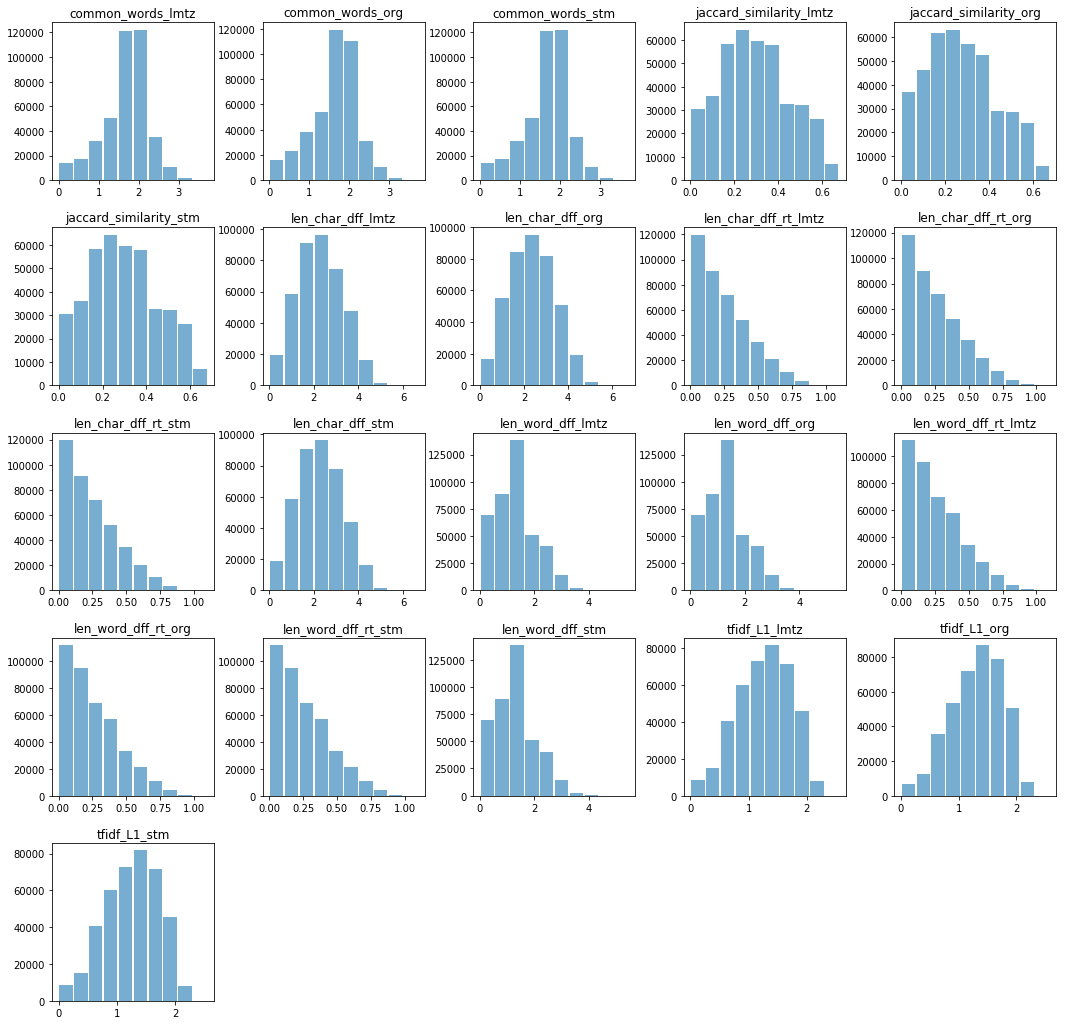
挖掘了三类共86个特征，如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 基本特征 | 原文本 | 首个单词是否相同 |
| 2 | 结尾单词是否相同 |
| 3 | 文本中性情感差异 |
| 4 | 文本正面情感差异 |
| 5 | 文本负面情感差异 |
| 6 | 文本相同情感差异 |
| 7 | fuzzy\_Qratio |
| 8 | fuzzy\_Wratio |
| 9 | fuzzy\_partial\_ratio |
| 10 | fuzzy\_partial\_token\_set\_ratio |
| 11 | fuzzy\_partial\_token\_sort\_ratio |
| 12 | fuzzy\_token\_set\_ratio |
| 13 | fuzzy\_token\_sort\_ratio |
| 14 | 句子对重复次数 |
| 15 | 句子1重复次数 |
| 16 | 句子2重复次数 |
| 17 | 平均重复次数 |
| 18 | 统计特征 | 单词个数差异 |
| 19 | 单词个数差异比率 |
| 20 | 字符个数差异 |
| 21 | 字符个数差异比率 |
| 22 | 相同单词数 |
| 23 | 相同单词数比率 |
| 24 | Jaccard相似度 |
| 25 | TFIDF-L1距离 |
| 26 | TFIDF-L2距离 |
| 27 | TFIDF-cosine距离 |
| 28 | stemming | 单词个数差异 |
| 29 | 单词个数差异比率 |
| 30 | 字符个数差异 |
| 31 | 字符个数差异比率 |
| 32 | 相同单词数 |
| 33 | 相同单词数比率 |
| 34 | Jaccard相似度 |
| 35 | TFIDF-L1距离 |
| 36 | TFIDF-L2距离 |
| 37 | TFIDF-cosine距离 |
| 38 | lemmatizing | 单词个数差异 |
| 39 | 单词个数差异比率 |
| 40 | 字符个数差异 |
| 41 | 字符个数差异比率 |
| 42 | 相同单词数 |
| 43 | 相同单词数比率 |
| 44 | Jaccard相似度 |
| 45 | TFIDF-L1距离 |
| 46 | TFIDF-L2距离 |
| 47 | TFIDF-cosine距离 |
| 48 | remove stop words | 相同单词数 |
| 49 | 相同单词数比率 |
| 50 | Jaccard相似度 |
| 51 | TFIDF-L1距离 |
| 52 | TFIDF-L2距离 |
| 53 | TFIDF-cosine距离 |
| 54 | embedding特征 | word2vec | word mover distance |
| 55 | normalized word mover distance |
| 56 | L1距离 |
| 57 | L2距离 |
| 58 | cosine距离 |
| 59 | canberra距离 |
| 60 | minkowski距离 |
| 61 | braycurtis距离 |
| 62 | 句子1向量的倾角 |
| 63 | 句子2向量的倾角 |
| 64 | 句子1向量的峰度 |
| 65 | 句子2向量的峰度 |
| 66 | GloVe | L1距离 |
| 67 | L2距离 |
| 68 | cosine距离 |
| 69 | canberra距离 |
| 70 | minkowski距离 |
| 71 | braycurtis距离 |
| 72 | 句子1向量的倾角 |
| 73 | 句子2向量的倾角 |
| 74 | 句子1向量的峰度 |
| 75 | 句子2向量的峰度 |
| 76 | universal sentence encoding | L1距离 |
| 77 | L2距离 |
| 78 | cosine距离 |
| 79 | 对数处理的cosine距离 |
| 80 | canberra距离 |
| 81 | minkowski距离 |
| 82 | braycurtis距离 |
| 83 | 句子1向量的倾角 |
| 84 | 句子2向量的倾角 |
| 85 | 句子1向量的峰度 |
| 86 | 句子2向量的峰度 |

其中一些特征存在较大的偏差，对其进行了对数化处理。部分特征未处理前的分布直方图如下示：



对数处理后如下图所示：



#### 模型对比

将训练数据中的20%作为验证数据，分别对比了随机森林和XGB两种模型以及基准模型在验证集的表现。

随机森林模型的参数采用默认值。XGB模型的参数来自[kaggle kernel](https://www.kaggle.com/benjaminkz/quora-question-pairs-xgboost/data)。

三种模型的训练时间和logloss如下图所示：

从图中可以看出，随机森林模型的训练时间最少，但logloss远远不能满足要求。XGB模型的训练时间远远小于BERT模型，并且logloss为0.4679，小于基准模型的0.4958。因此，选择XGB模型处理本项目。

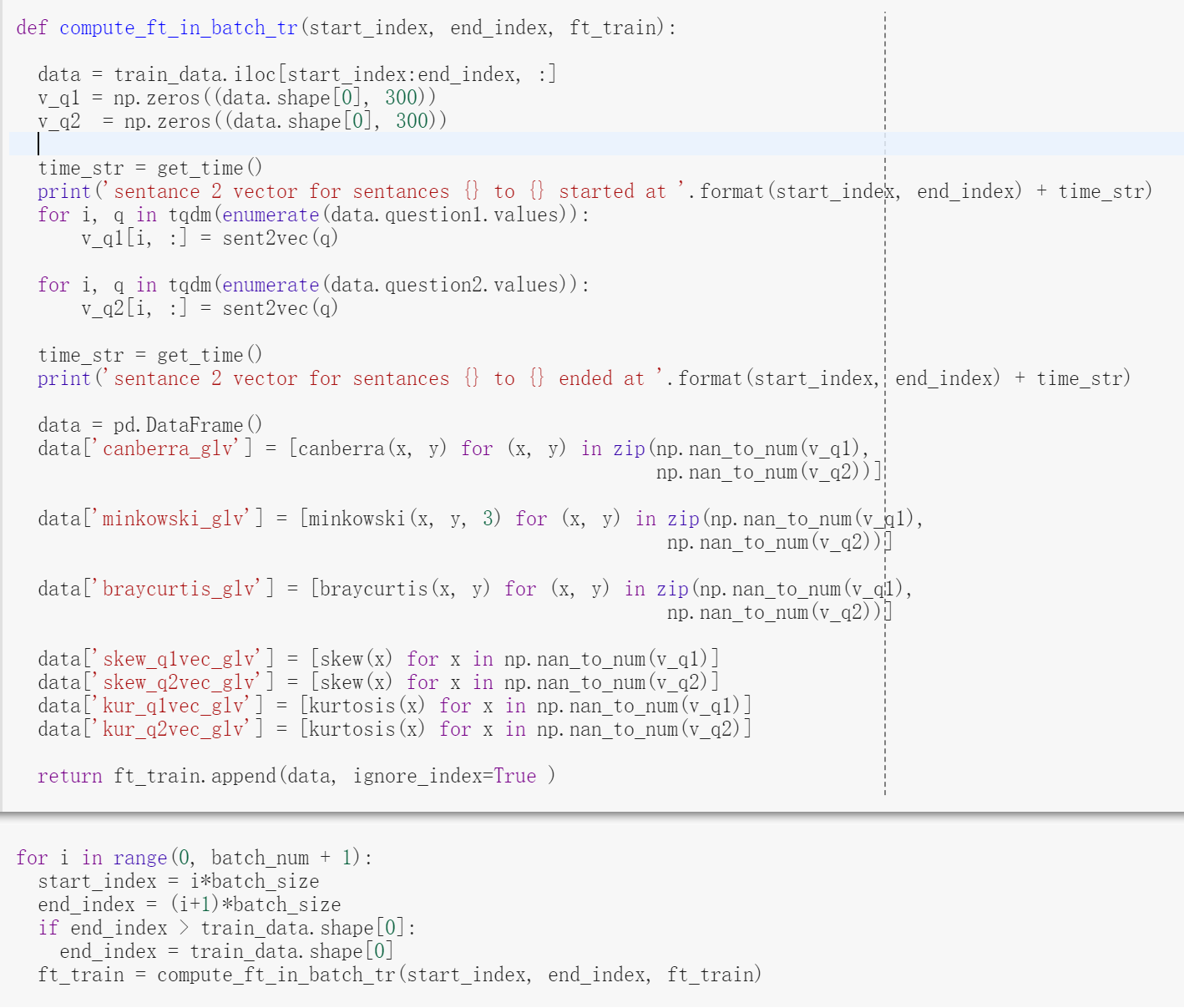
#### 训练难点

1. **测试集过大**

这部分面临的最大问题是测试集数据较大。计算测试集embedding特征时，常常会由于出现很大的矩阵而导致超过内存使用量。解决方法是分批处理测试集，减小了每次处理的矩阵维度。

代码如下：





1. **Logloss过大**

由于前期采用的参数来自[kaggle kernel](https://www.kaggle.com/benjaminkz/quora-question-pairs-xgboost/data)，因此需要对参数进行调节。

但是在在调节参数时发现，logloss总是无法通过调节参数达到要求的logloss。经过搜集资料发现，对模型表现贡献最大的是特征以及特征数量。因此收集了更多的特征计算方法，扩充了特征的数量。主要包括增加了基本特征、word embedding增加了word2vec。

下图示意了特征数量增加后，XGB在学习率为0.1情况下的表现：

1. **参数调节**

XGB模型在许多分类问题都有不错的表现。使用该方法的难点在于确定合适的参数组合。

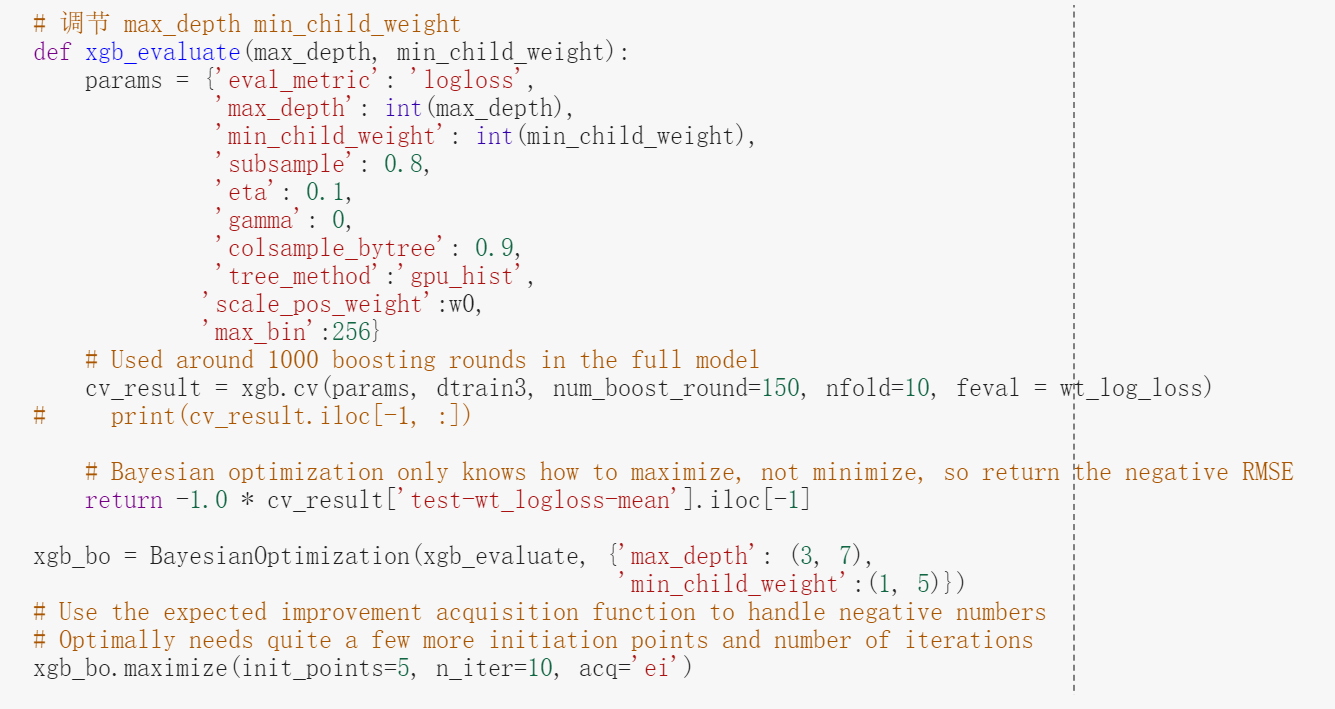
该模型需要调节的参数包括学习率（eta）、基学习器数量、最大树深度、最小叶子节点样本权重和、gamma（节点分裂所需的最小损失函数下降值）、subsample（控制对于每棵树，随机采样的比例）和colsample\_bytree（控制每棵随机采样的列数的占比）。

通常的参数调节方法为grid search，即历遍所有可能的组合，分别训练模型，选择最佳模型。然而这种方法效率太低，耗时。更常用的是随机的grid search，即随机产生大量可能的组合，选择最佳模型。这种方法则有可能无法选到最优的参数组合，并且时间成本较高。同时，XGB的sklearn 包不支持GPU加速的grid search，因此时间成本更大。

另外还有贝叶斯优化算法，该算法假定寻找最优化参数的过程是一个高斯过程。通过随机遍历一定的数据点并拿到结果之后，可以大致绘制出整个数据的分布曲线，从而确定最优化参数。

考虑到本项目的时间成本，决定采用贝叶斯优化算法。使用的库是bayes\_opt。

代码如下，参考自[kaggle kernel](https://www.kaggle.com/btyuhas/bayesian-optimization-with-xgboost)：



1. **正负例比率与指标计算方法**

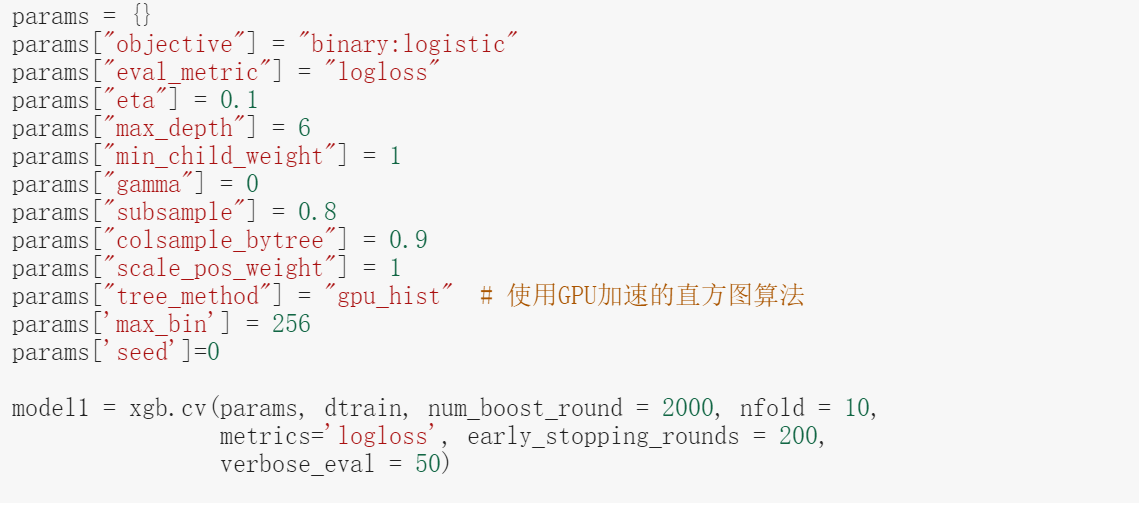
首次将预测结果提交至kaggle时发现，kaggle的logloss得分与模型计算结果相差较大。

经过浏览讨论版发现，测试集与训练集的正负例比率相差较大。如果按照训练集的正负例比率训练模型，模型在测试集的表现与训练集不同。因此需要通过scale-pos-weight参数进行调整。

另外，kaggle使用的是加权的计算方法。为了在调整参数时获得与评分接近的结果，需要使用加权的logloss计算方法来评估调参时的模型表现。项目中采用了[kaggle kernel](https://www.kaggle.com/benjaminkz/quora-question-pairs-xgboost/data)分享的加权计算方法。

### 完善

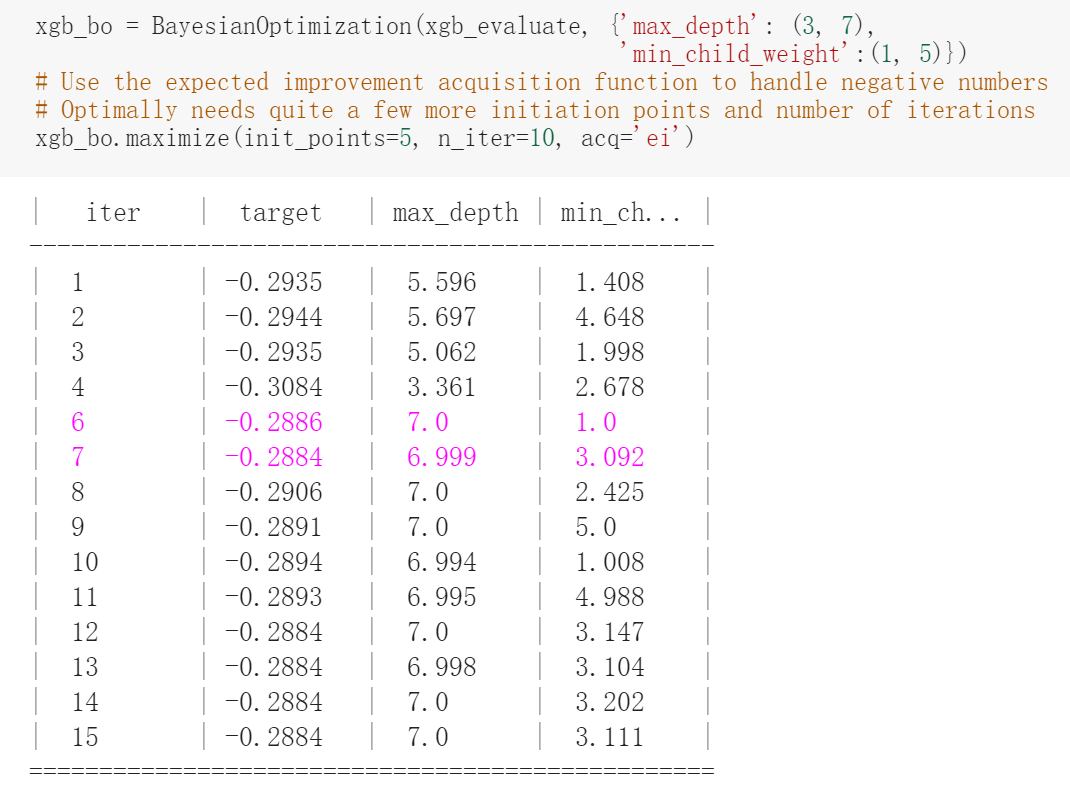
首先，在学习率为0.1的情况下确定基学习器的数量。使用xgb.cv方法。参数及运行结果如下：





在基学习器数达到950之后，test-logloss开始上升，因此在0.1的学习率下，基学习器数量为 较为合适。不过考虑到后续需要调整多个参数，因此选择学习率0.1，基学习器 150的组合进行后续调参。

接着调节max-depth 和min-sample-weight的组合。这两个参数的一搬取值为1-6。采用贝叶斯优化算法进行调参。参数设置和调参结果如下所示：



从上图可以看出，max-depth 和min-sample-weight的组合在6和3的情况下达到最小logloss。max-depth 和min-sample-weight的组合优化6为 3和 。

接着，调节gamma。该数值影响模型的拟合情况，过小可能导致过拟合。过大可能导致欠拟合。采用贝叶斯优化算法进行调参。参数设置和调参结果如下所示：



从上图可以看出，其他取值的gama都没有达到比0更小的logloss，因此维持gamma = 0。

最后，调节subsample和colsample\_bytree组合。这两个参数的一般取值在0.5之0.9之间。采用贝叶斯优化算法进行调参。参数设置和调参结果如下所示：



从上图可以看出，subsample和colsample\_bytree组合在 0.99997和 0.6009的情况下达到最小logloss。subsample和colsample\_bytree组合优化为0.99997和 0.6009。

确定这些参数后，参考相关文章和项目的经验，选择了0.02的学习率和3600的基学习器组合。

## 结果

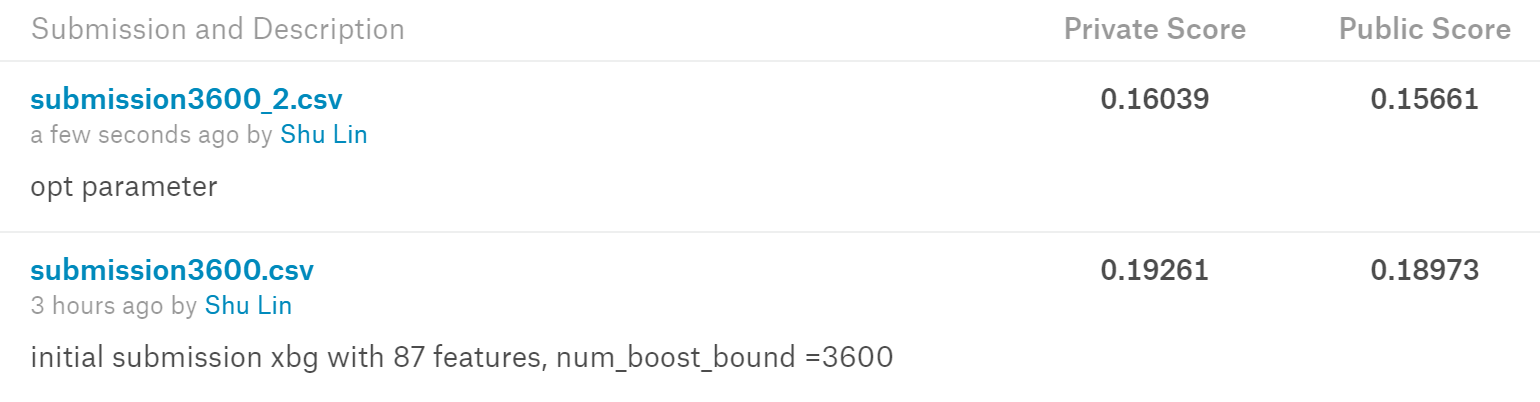
### 模型评价与验证

最终选用的模型参数如图所示：



训练时间为1382s，预测时间为16s。

将预测结果提交至kaggle进行评分。完善前与完善后的得分如下所示。两次提交的模型主要有三点不同。一是去掉了‘qid’这一特征；二是使用调整正负例比率训练模型；三是参数的优化。这三点变化对结果有了显著的提高。由于时间的关系没法进一步研究。



该成绩在private leaderboard 上是332/3307，达到了前10%。由此，可以认为该模型有了不错的表现。

## 项目结论

### 对项目的思考

我原本以为这个项目比较简单，业界已有成熟、简单、有效的解决方案。

实际尝试才发现，许多宣称有效的新方法在实际中表现并不尽如人意。表现优秀的解决方案常常是多种看似简单的方法的堆叠、平均。

比赛中排名前列的团队，大多尝试了非常多的方法，最终选择了几种进行组合。这样的解法，看似只花费了力气，实际上仍然是了不起的工作。很多时候，花费很多力气的方法就是最好的方法。

在完成中，我学到了很多实用的技巧。例如使用pickle保存生成的语言模型、训练模型等等，在需要的时候能更快地使用。

### 需要作出的改进

本项目时间比较仓促，准备不够充分，所以整个流程不够完善。例如开题报告确定的方案没有用上。原本计划主要采用BERT模型完成项目，但是由于训练时间的关系放弃了。

在实施过程中也遇到了很多想不到的困难。这些困得都只有动手做才会发现，也由此获得了解决方法，也算因祸得福。

本项目的特征工程部分实际上还有提高的空间。讨论版上分享了许多句子空间关系的特征，由于时间关系没有尝试。

最后，同样由于时间关系，没有调节学习率与基学习器的组合。这方面也有优化的空间。

## 参考资料

1. Natural language processing<https://en.wikipedia.org/wiki/Natural_language_processing>
2. 自然语言处理-简介 <https://www.zhihu.com/topic/19560026/intro>
3. Is That a Duplicate Quora Question? <https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur/>
4. A Gentle Introduction to the Bag-of-Words Model <https://machinelearningmastery.com/gentle-introduction-bag-words-model/>
5. What Are Word Embeddings for Text? <https://machinelearningmastery.com/what-are-word-embeddings/>
6. Gentle Introduction to Statistical Language Modeling and Neural Language Models <https://machinelearningmastery.com/statistical-language-modeling-and-neural-language-models/>
7. 文本主题模型之LDA(一) LDA基础 <https://www.cnblogs.com/pinard/p/6831308.html>
8. 自然语言处理基础与实战（8）- 主题模型LDA理解与应用 <https://www.jianshu.com/p/74ec7d5f6821>
9. 从0到1，了解NLP中的文本相似度 <https://www.jianshu.com/p/22afb6d25e74?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation>
10. 1st place solution <https://www.kaggle.com/c/quora-question-pairs/discussion/34355#latest-572705>
11. BNP Paribas Cardif Claims Management, Evaluation <https://www.kaggle.com/c/bnp-paribas-cardif-claims-management/overview/evaluation>
12. Log Loss Evaluation Section Missing. <https://www.kaggle.com/c/quora-question-pairs/discussion/30605#latest-172575>
13. Manhattan LSTM model for text similarity <https://medium.com/@gautam.karmakar/manhattan-lstm-model-for-text-similarity-2351f80d72f1>
14. BERT <https://github.com/google-research/bert>
15. BERT和ULMFIT embedding比较文本分类结果 <https://www.cnblogs.com/demo-deng/p/10797405.html>
16. Bert-as-service <https://github.com/hanxiao/bert-as-service>
17. Predicting Movie Review Sentiment with BERT on TF Hub <https://github.com/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb>
18. Complete Guide to Parameter Tuning in XGBoost with codes in Python <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>
19. quora-question-pairs-xgboost <https://www.kaggle.com/benjaminkz/quora-question-pairs-xgboost/notebook>
20. Bayesian Optimization with XGBoost <https://www.kaggle.com/btyuhas/bayesian-optimization-with-xgboost>