机器学习纳米学位

毕业项目： Quora 句子相似度匹配

张培军 优达学城

2020 年 2 月 20 日

I. 问题的定义

Quora 是一个问答网站，在这里用户可以选择性的提出、回答、编辑和组织各种问题，也是一个可以获取及分享各种知识的网络平台。它可以让世界各地拥有独特视野和睿智答案的人们联系在一起，促进了人们之间的相互学习，以及能够更好地认识世界。每个月将有超过 1 亿用户访问 Quora，基于如此巨大的访问量，一定有很多人提出重复的问题，或者这些问题包含相同的内容。然而这些具有相似意图的问题可能会使得寻求者需要花费更多的时间，才能找到所需的最佳答案，也使得答题者感到必须要回答很多内容相似的问题。

Quora 很重视所提的问题的规范性，为提升活跃用户对网站的使用体验，并长期为他们提供更多的价值，采用相关先进的技术手段提炼发现内容相似的问题。

Kaggle 举办了相关竞赛： Quora Question Pairs。主要目的是，通过应用先进的技术来判断两个问题内容含义是否相似。通过挑战解决这一自然语言处理问题，进一步提升 Quora 用户的使用体验。

问题陈述

数据集是 Quora 于 2017 年公开的句子匹配数据集，其通过给定两个句子的一致性标签标注，判断句子是否一致。

Quora 数据集训练集共包含 400K 的句子对，而测试集包含大约有 200 多万句子对，可以从 Kaggle 网站进行下载。

	Train	Test
Data Size	404290	2345796
Vocab Size	95603	101049

Kaggle 端的数据集，其由 Train,Test 两部分构成，通过在 Train 数据集上进行验证集划分、建模，在 Test 数据集上进行测试，并且提交到 Kaggle 进行测评。最终的目标是最小化句子相似度的预测值在测试集上的 logloss 值。

训练集：255027 条非相似句子对和 149263 相似句子对。

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
4	4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?	I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1

测试集：2345796 条句子对，没有标签标注。

	test_id	question1	question2
0	0	How does the Surface Pro himself 4 compare with iPad Pro?	Why did Microsoft choose core m3 and not core i3 home Surface Pro 4?
1	1	Should I have a hair transplant at age 24? How much would it cost?	How much cost does hair transplant require?
2	2	What but is the best way to send money from China to the US?	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?
5	5	How are the two wheeler insurance from Bharti Axa insurance?	I admire I am considering of buying insurance from them

本文选用 Xgboost 算法模型，并结合数据完善算法。由于算力和时间限制，算法涉及的特征集包含只选取了 45 个特征，最后通过验证，模型取得了较好的效果，满足了项目要求。所有特征从总体上可以分为 nlp 和 leaky 两类特征。

评价指标

本项目的评价标准用的是 LogLoss：

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

这个损失函数需要针对每行记录计算预测出问题是否相似的概率值，而不是 0-1 二分值。它的特点就是相比准确率这个指标来说能够更好标识预测的精度，这个损失函数值随着预测精度的提高而减少。

II. 分析

数据探索及可视化

在开始项目后，我们要先对数据进行分析。由于数据量很大，因此，我们需要使用一些数据分析工具以及可视化工具，如 pandas,matplotlib, seaborn 等。对数据做初步分析非常具有意义，比如对分类问题中的特征，观察数据的分布，可以直观的看出这个特征是否有足够的"区分度"。 一个直观的可视化展示也非常有助于进行思考和挖掘数据分布上特征。

项目数据集分为训练集和测试集。数集中的每项数据包含需要确认是否相似的两个问题。另外，训练集还包括一些额外的列特征：其中一个标注两个问题是否相似，另外两个是每个问题各自的 id。

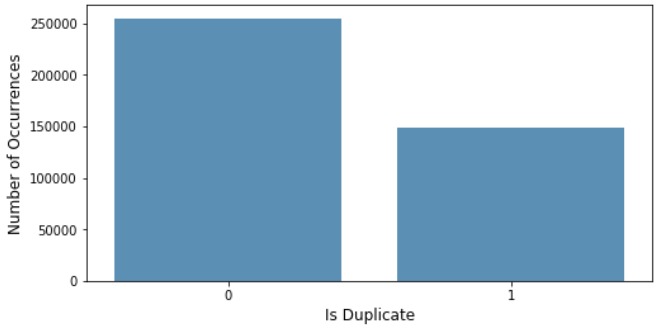
训练数据中各特征的描述：

qid1, qid2——每个句子各自唯一的 id 值

question1, question2——每个句子的具体内容

is_duplicate ——目标变量，如果 question1 和 question2 表达完全相同的内容则标注为 1；否则标注为 0.

为分析句子对的相似性，下面对数据集进行进一步探索。首先观察目标变量的分布。




```
is_dup / is_dup.sum()
```

```
0    0.630802
1    0.369198
Name: is_duplicate, dtype: float64
```

由此可知，在训练集中，非相似句子对占 63%，相似句子对占 37%。在很多文献中提到，训练集和测试集之间表现的分类平衡是不同的（训练集有 37%的正例，而测试集只有大约 16.5%）。为了接近测试集的数据分布，采取的方法是对训练数据再平衡，对训练数据中的正例进行欠采样，同时对负例进行过采样。

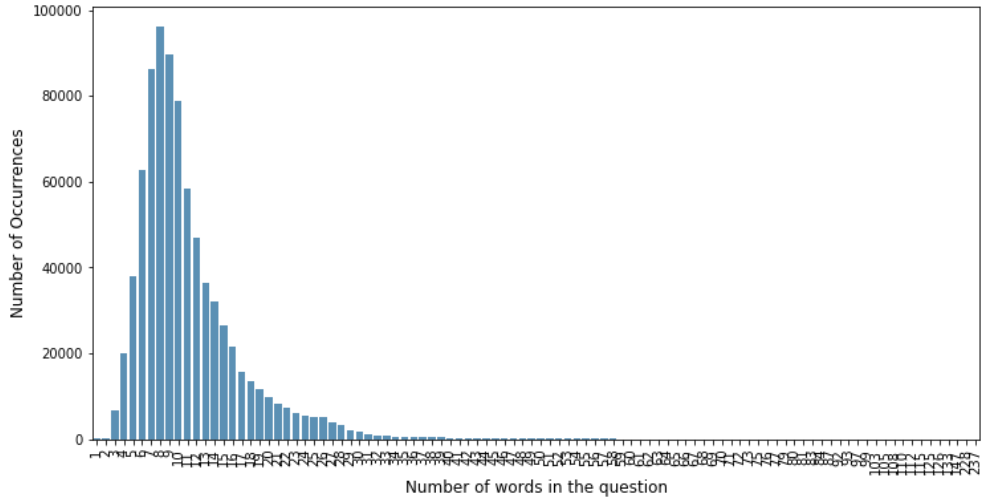
下面对训练集中的句子内容进行探索，首先分析句子包含单词数量的分布情况。

训练集词表数量: 95596

测试集词表数量: 101312

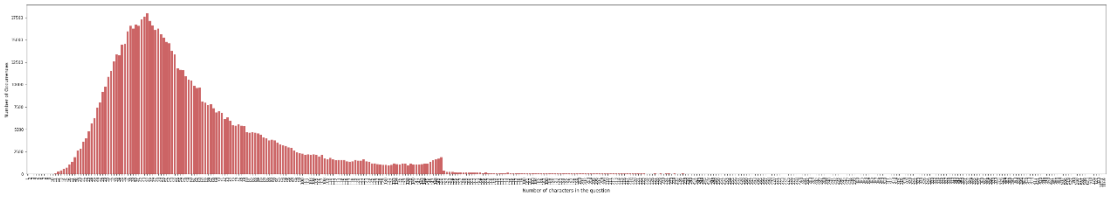
测试集中有多少词汇不在训练集里 / oov（out of vocab): 41446

- 单词数量的分布



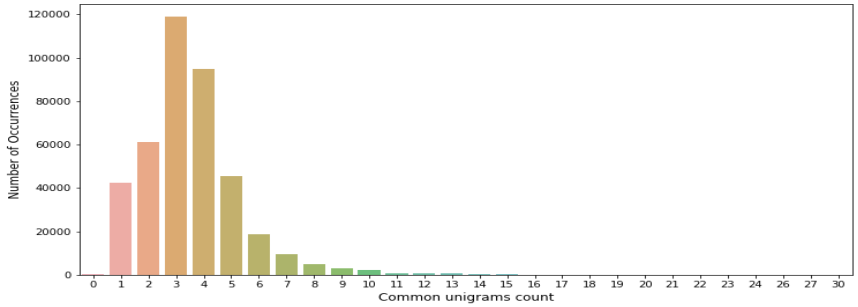
此图呈现为右偏态分布，从右偏态的延伸可看出，句子最多包含 237 个单词。包 1 个和 2 个单词的句子也比较少。

- 查看字符数量分布



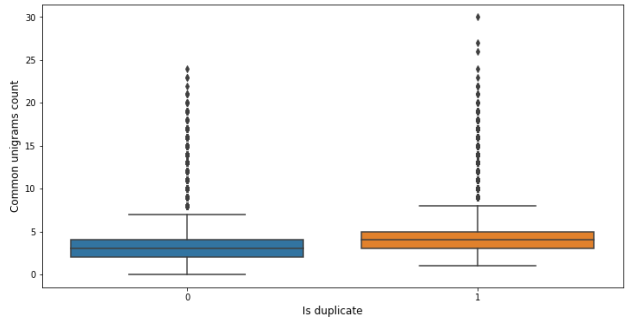
字符数量分布也是右偏态，可以发现，由于 Quora 之后规定发布问题的最大字符数为 150，所以在图中横轴数字 150 处出现了数量的明显下降。

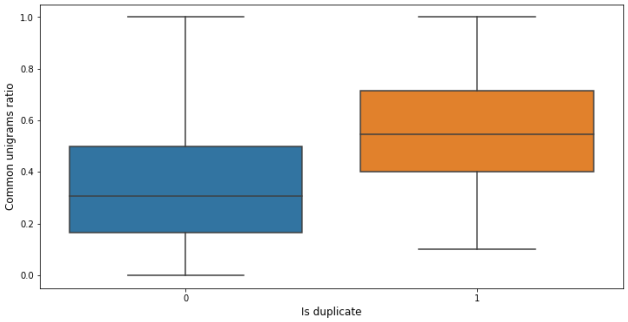
- 共享 unigram 分布



由图可知，句子之间没有共享 unigram 的句子对非常少

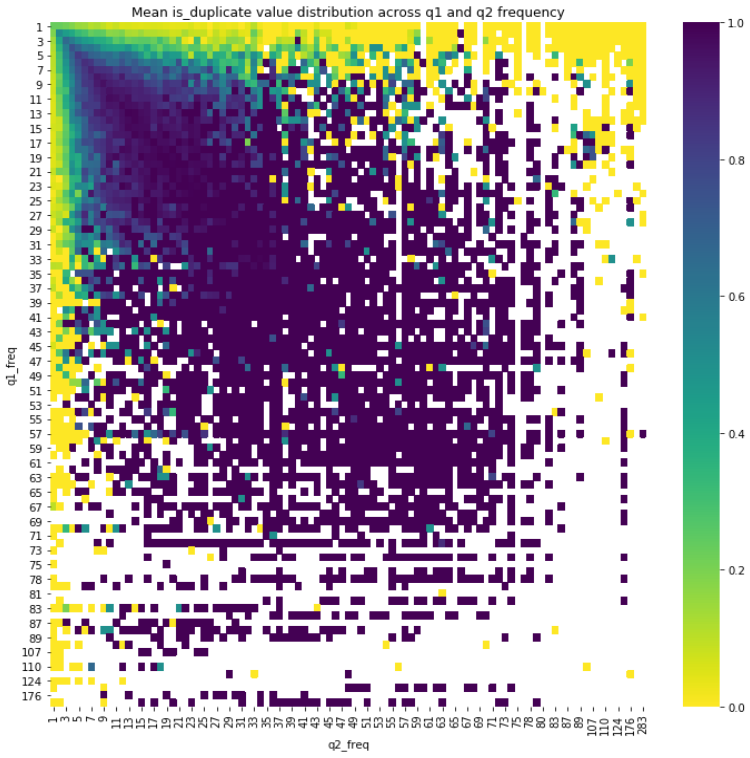
对于目标变量中 0 和 1 两类分类数集，问题对中共享 unigram 的数量表现出的特性有着明显的区别，可以做为特征参与模型。



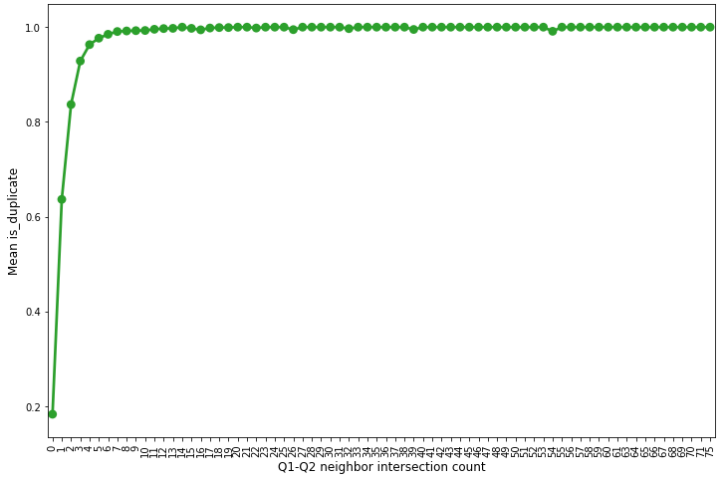


- **Leaky Features**

有 kagglers 分享了他们发现的 leaky feature，对算法模型有较好的提升效果。



1. Frequency based feature by Jared Turkewitz
2. Intersection of common neighbors by Krzysztof Dziedzic implemented by tour1st



可见，q1_q2_intersect 的值越大，对应的问题对的集合 is_duplicate 的均值越接近于 1。特征 q1_q2_intersect 、q1_freq 、q2_freq 与目标变量 is_duplicate 有较强的相关性，有利于模型做出较好的预测。

算法和技术

Xgboost 算法利用梯度提升算法框架，是一种基于决策树的集成机器学习算法，现在变的越来越流行，尤其是在 kaggle 项目。由于已经在许多的大赛中取得了较好的成绩并且应用广泛，至今 Xgboost 算法仍然热度不减。它设计并优化了提升树算法。相比深度学习算法，在处理小型到中型结构化和表格化数据、解决分类问题时，具有一定的优势。

Xgboost 算法的目标函数 objective function 如下所示，包含训练损失项和正则化项：

$$\text{obj} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \Omega\left(f_i\right)$$

由于包含树的模型不适合直接对整体进行优化，因而采用加法学习方法，每次学习当前的书，找到当前最佳的树模型加入到整体模型中。

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1\left(x_i\right) = \hat{y}_i^{(0)} + f_1\left(x_i\right) \\
\hat{y}_i^{(2)} &= f_1\left(x_i\right) + f_2\left(x_i\right) = \hat{y}_i^{(1)} + f_2\left(x_i\right) \\
&\cdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k\left(x_i\right) = \hat{y}_i^{(t-1)} + f_t\left(x_i\right)
\end{aligned}$$

由此：

$$\begin{aligned}\text{obj}^{(t)} &= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{i=1}^{t} \Omega\left(f_i\right) \\ &= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(x_i\right)\right) + \Omega\left(f_t\right) + \text{constant}\end{aligned}$$

上式通过泰勒展开，保留二次项，可得：

$$\text{obj}^{(t)} = \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t\left(x_i\right) + \frac{1}{2} h_i f_t^2\left(x_i\right)\right] + \Omega\left(f_t\right) + \text{constant}$$

其中：

$$\begin{aligned}g_i &= \partial_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right) \\ h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)\end{aligned}$$

移除所有的常数项，在第 t 步的目标函数变成：

$$\sum_{i=1}^{n} \left[g_i f_t\left(x_i\right) + \frac{1}{2} h_i f_t^2\left(x_i\right)\right] + \Omega\left(f_t\right)$$

定义树函数和正则化项：

$$f_t\left(x\right) = w_{q(x)}, w \in R^T, q : R^d \rightarrow \{1, 2, \cdots, T\}.$$

$$\Omega\left(f\right) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

此时，t 层树模型的目标值可推导如下：

$$\begin{aligned}\text{obj}^{(t)} &\approx \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T\end{aligned}$$

其中，$I_j = \{i | q\left(x_i\right) = j\}$ 是映射第 j 叶的数据点的所有标示下标的集合。进一步压缩上式可得：

$$\text{obj}^{(t)} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} \left(H_j + \lambda\right) w_j^2\right] + \gamma T$$

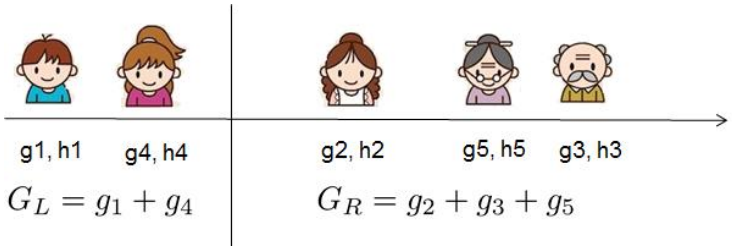
其中：

$$\begin{aligned}G_j &= \sum_{i \in I_j} \\ H_j &= \sum_{i \in I_j} h_i\end{aligned}$$

由于上式为一元二次式，可得最优解：

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

$$\text{obj}^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$



对于某层叶子，将特征按一定的顺利排序，从左至右依次扫描切分，最终利用增益 Gain 值决定是否分裂或最佳切分点。

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$

相比 GBDT，Xgboost 算法主要有以下特点：

1、传统 GBDT 以 CART 作为基分类器，xgboost 还支持线性分类器，这个时候 xgboost 相当于带 L1 和 L2 正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题）。

2、传统 GBDT 在优化时只用到一阶导数信息，xgboost 则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数。

3、Xgboost 在代价函数里加入了正则项，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的 score 的 L2 模的平方和。从 Bias-variance tradeoff 角度来讲，正则项降低了模型的 variance，使学习出来的模型更加简单，防止过拟合，这也是 xgboost 优于传统 GBDT 的一个特性。

4、列抽样（column subsampling）。xgboost 借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算，这也是 xgboost 异于传统 gbdt 的一个特性。

本项目借鉴以往的经验和项目信息，在代码中引入 Xgboost 模型时，只调整如下参数：

params = {

'objective': 定义最小化损失函数类型

'eval_metric': 验证数据的评价标准

'eta': shrinkage 参数，用于更新叶子节点权重时，乘以该系数，避免步长过大。参数值越大，越可能无法收敛。把学习率 eta 设置的小一些，小学习率可以使得后面的学习更加仔细。

'max_depth': 每颗树的最大深度，树高越深，越容易过拟合
'subsample': 样本随机采样，较低的值使得算法更加保守，防止过拟合，但是太小的值也会造成欠拟合。
'base_score': 所有实例的初始预测分数
}

在自然语言处理领域，面对计算句子相似度问题时，若将获取的文本数据转换成输入模型的文本特征，通常主要解决两个问题：首先，需要选择一个文本的表示方式。其次，需要选择一个衡量文本的相似度的距离度量。

词向量是用一个向量来表示某个词的方法。本项目主要涉及到的两种这类文本表示方式，通过这些表示方式把文本符号数学化：

- TF-IDF 词频-逆文档频率

TF-IDF 中，每一单词对应的值是词频和逆文本频率的乘积。TF 基于单个文本计算，IDF 则是基于整个语料库计算。词频：是指某个词在文本中出现的次数 。逆文档频率：如果一个单词在整个语料库中的很多文本里都出现，那么该单词的逆文本频率就越小。TF-IDF 与文本中的某个词的出现次数成正比，与整个语料库中存在该词的文本的总数量成反比。TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。一个文本中的某个单词的 TF-IDF 值越大，说明在给定语料库的前提下该单词在这个文本中越重要。最后，它的缺点：一方面，单纯以"词频"衡量一个词的重要性不够全面，有时重要的单词可能出现次数并不多；另一方面，TF-IDF 也忽略文本词序、语法和句法。

- Word2vec 模型

对于 TF-IDF 这种基于词袋模型来说，不同文本得到的词向量都近乎正交，而且无法计算独立单词组成的文本的距离。word2vec 只能得到词向量，比较词之间的相似度，通过简单的加权、tag 加权、tf-idf 加权等方式得到文档向量。

距离度量：

- cosine distance

$$Cosine(\mathbf{a},\mathbf{b}) = \frac{<\mathbf{a},\mathbf{b}>}{|\mathbf{a}| \times |\mathbf{b}|}$$

衡量的是空间向量的夹角，更体现方向上的差距，对绝对数值不敏感，修正了度量标准不统一的问题。由于余弦距离对绝对数值不敏感，所以可以通过所有的样本在维度上减去一个均值（即令特征的均值为 0）来修正这种不合理。

- cityblock distance

在欧几里得空间的固定直角坐标系上两点所形成的线段对轴产生的投影的距离总和。例如在平面上，坐标（x1, y1）的点 P1 与坐标（x2, y2）的点 P2 的曼哈顿距离为：$|x_1 - x_2| + |y_1 - y_2|$

- canberra distance

堪培拉距离 (Canberra Distance)被认为是曼哈顿距离的加权版本。
在真实 n 维向量空间可表示为：

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

其中 **p**、**q** 表示两个向量。

- minkowski distance

明氏距离又叫做明可夫斯基距离，是欧氏空间中的一种测度，被看做是欧氏距离和曼哈顿距离的一种推广。

对于两点 $P = (x_1, x_2, \ldots, x_n)$ and $Q = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$ 之间的明氏距离公式为：

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}.$$

- braycurtis distance

Bray Curtis 距离主要用于生态学和环境科学，计算坐标之间的距离。该距离取值在[0,1]之间。它也可以用来计算样本之间的差异。

$$d_{ij} = \frac{\sum_{k=1}^{n} \left| x_{ik} - y_{jk} \right|}{\sum_{k=1}^{n} x_{jk} + \sum_{k=1}^{n} y_{jk}}$$

- 编辑距离

编辑距离是指两个文本之间，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 / 单词替换成另一个字符 / 单词，插入一个字符 / 单词，删除一个字符 / 单词。

- Word Mover's Distance 词移距离

简单来说，对于一个文本中的所有词的词向量，分别找出另一个文本中距离和该词向量距离最近的词向量（即最小距离），最后将这些最小距离求和。不过实际中，在词移距离中的词向量与词向量之间并不是一对一的关系。一个文本中词向量可能对应着另一个文本中的多个词向量，只是权重不同。

$$\min_{T\geqslant 0}\sum_{i,j=1}^{n}T_{ij}||\mathbf{x_i}-\mathbf{x}_j||$$

$$s.t.\sum_{j=1}^{n}T_{ij}=d_i,\quad \sum_{i=1}^{n}T_{ij}=d_j$$

除此之外，本文还对文本的统计特征进行了挖掘，比如句子长度，词长度，共享词数量等，这些也是最直观的度量方法。

基准模型

衡量模型 Xgboost 性能，可以建立一个基准模型，输出基准数值。由于模型最后的预测值是概率，取实际测试集所有标签值的均值做为结果进行提交。把最后的得分做为基准值。

训练集所有的正例的比例是 36.9%，取常数 0.369 做为所有测试集的预测值进行提交，得到 log loss 值或者 public LB score 0.554。

$$r=\frac{logloss+log(1-p)}{log\left(\frac{1-p}{p}\right)}$$

由上式可得 r=0.174，其中 r 代表参与计算 log loss 的数据测试集中正例所占的比例。0.554 是在 public LB 上的得分，包含的测试数据占总数据集的 6%，由于和总测试集的正例比例接近，可由此结果近似整个测试集的正例比例。最后将常数 0.174 做为每个测试数据的预测值进行提交，得分 score 0.463，将此数值做为基准值。

III. 方法

数据预处理

特征工程是构建良好机器学习模型的重要组成部分，特征是决定效果最关键的一环。引用一句话“数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限”。本文应用多种方法从文本数据中提取 45 个特征，主要分为 nlp 特征和 leaky 特征两大类。

nlp 特征

- Basic Features：

Feature	Description
len_q	句子中的字符数量，包括空格
len_char_q	句子中的字符数量，不包括空格
diff_len	句子对之间的字符数量差
char_diff_unq_stop	筛选停止词后句子对之间的字符数量差
char_ratio	句子对之间字符数量之比
len_word_q	句子中的单词数量
wc_diff	句子对之间的单词数量差
wc_diff_unique	句子对之间唯一词的数量差
wc_diff_unq_stop	筛选停止词后句子对之间唯一词的数量差
wc_ratio	句子对之间单词数量之比
wc_unique_ratio	句子对之间唯一词数量之比
wc_ratio_unique_stop	筛选停止词后句子之间唯一词数量之比
total_unique_words	每个句子对中唯一词的数量
total_unique_words_stop	筛选停止词后每个句子对中唯一词的数量
common_words	每个句子对中的共享单词数
word_match	句子对中共享词的数量在筛选停止词后句子对中单词总数的占比
Jaccard	共享词的数量在句子对中总单词数的占比
same_start	如果句子对有相同的 start word 就返回值 1，否则返回 0

- TF-IDF Features：

tf-idf 用于测量句子中单词的重要性。重要性由句子 （tf） 中的词频表示，并由词语集（idf）中的单词频率补充。

Feature	Description
tfidf_wm	tfidf 转化后的 word match
tfidf_wm_stops	tfidf 转化和筛选停止词后的 word match

- Word2vec features：

Word2vec 是一个双层神经网络模型，用于生成单词嵌入。Word2vec 以大量文本作为输入，并生成数百个维度的矢量空间。具有类似含义的单词在其矢量空间中距离很近。项目导入了谷歌预训练的 Word2vec 模型，并使用 gensim 包在 python 上运行模型。它输出数据中每个单词的矢量。然后，计算了测量向量之间的相似性的距离要素、测量分布形状的偏斜度和峰度特征。

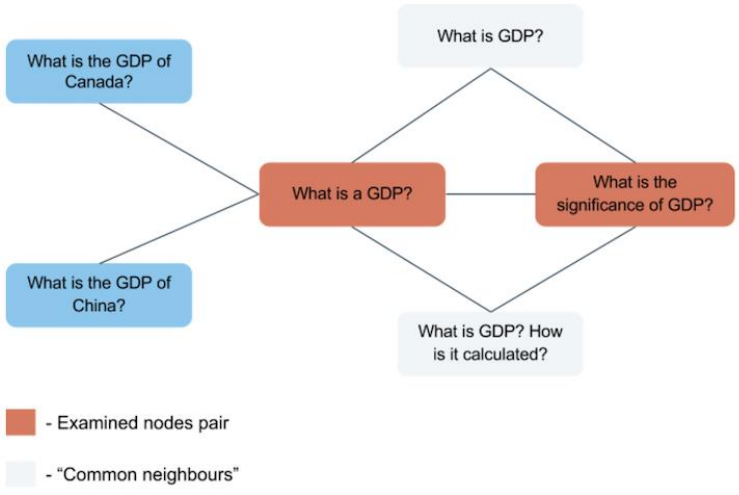
Feature	Description
wmd	word mover distance
norm_wmd	normalized word mover distance
cosine_distance	cosine distance
cityblock_distance	cityblock distance
canberra_distance	canberra distance
minkowski_distance	minkowski distance
braycurtis_distance	braycurtis distance
skew_q1vec	question1 词向量的偏斜度
skew_q2vec	question2 词向量的偏斜度
kur_q1vec	question1 词向量的峰度
kur_q2vec	question2 词向量的峰度

- Fuzzywuzzy Features：

fuzzywuzzy 是 python 的一个用来进行模糊字符串匹配的工具,它的基本原理是计算两个相似字符串之间的编辑距离，fuzzywuzzy 提供了多种比率(ratio)来描述两个字符串之间的相似程度。主要包含以下几种比率特征： QRatio、WRatio、Partial ratio、Partial token set ratio、Partial token sort ratio、Token set ratio、Token sort ratio

Leaky 特征：

Leaky features 引起了很多争论。因为这些要素不是有意义的 NLP 要素，虽然在竞赛中特征效果很好。但在实际项目中并不实用，因为我们有可能从实际测试数据获取不到有意义的的 leaky 特征。但由于项目已有数据集的形成方式，它产生了数据集中的一些模式，可以从中提取出来。令人怀疑这些特征是



否会在真实场景中发挥作用，但在纯粹为改进模型效果的的背景下，它们确实能够提高预测评分。

为了使用这种有价值的信息，将数据集作为图形显示起来很方便。它可以通过多种方式完成。例如，让我们构建一个图形，其中每项记录都将用两个连接边缘的节点表示。同时，一个节点对应于数据集中的一个问题。

如图所示，q1_q2_intersect 计算的就是与句子对"What is a GDP?,What is the significance of GDP"中的问题都组成过句子对的"Common neighbours"的数量。

Feature	Description
q1_frequency	question1 在数据集中出现的次数
q2_frequency	question2 在数据集中出现的次数
q1_q2_intersect	与 question1 和 question2 都组成过句子对的句子数量

执行过程

Xgboost 算法是任何愿意在竞赛中获得较好分数的参与者不可或缺的方法。由于受到计算能力和时间的限制，我的方法只采用了一个单一的模型解决方案。将之前提取的 45 个特征做为模型的输入。

模型调参步骤：1、根据经验，选出对模型效果影响较大的超参。2、按照经验设置超参的搜索空间，比如学习率的搜索空间为[0.001，0.1]。3、选择搜索算法，比如 Random Search、Grid Search 和一些启发式搜索的方法。4、验证模型的泛化能力。

要想模型的表现有大幅的提升，在调整模型参数也必须清楚，仅仅靠部分参数调整和模型小幅优化，想要让模型表现就有大幅提升是基本不可能的。要想模型在表现上有质的飞跃，还需要依靠其他手段。比如，特征工程(feature egineering) ，模型融合(blending)以及堆叠(stacking)等。

由此在基于大量特征工程工作基础上，本项目未把调参工作做为工作重点。参考经验和他人做法，设定模型参数：

```
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 7
params['subsample'] = 0.6
params['base_score'] = 0.2
```

在测试集的标签未知的情况下，我们需要自己构造测试数据来验证模型的泛化能力以及鲁棒性，因此把训练集分割成训练集和验证集两部分，训练集用于训

练，验证集用于验证。然后，在训练数据集上对分类器进行训练，本项目划分的该数据集占总数据集的 90%，并在 10% 的验证集上进行测试。

```
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.1, random_state=4242)
```

在进行模型训练之前，还有一个值得关注的点。训练集和需预测评分的测试集的分类不平衡不一致，具体来说，在训练集中，大约 37%是阳性标签，而在测试集中，大约在 16.5%左右。由于损失函数 logloss 对不平衡很敏感，对验证验证集使用多个折数，在验证集上获得的结果可能不会反映在 Kaggle leaderboard（LB） 上。因此，为了在 LB 上获得较好的结果，对拆分后的训练集和验证集中的正例进行一些欠采样，同时对负例进行过采样，最终正例比例达到约 19%

```
pos_train = X_train[y_train == 1]
neg_train = X_train[y_train == 0]
X_train = pd.concat((neg_train, pos_train.iloc[:int(0.8*len(pos_train))], neg_train))
y_train = np.array([0] * neg_train.shape[0] + [1] * pos_train.iloc[:int(0.8*len(pos_train))].shape[0] + [0] * neg_train.shape[0])
print(np.mean(y_train))
del pos_train, neg_train
```

```
0.18975293212164862
```

```
pos_valid = X_valid[y_valid == 1]
neg_valid = X_valid[y_valid == 0]
X_valid = pd.concat((neg_valid, pos_valid.iloc[:int(0.8 * len(pos_valid))], neg_valid))
y_valid = np.array([0] * neg_valid.shape[0] + [1] * pos_valid.iloc[:int(0.8 * len(pos_valid))].shape[0] + [0] * neg_valid.shape[0])
print(np.mean(y_valid))
del pos_valid, neg_valid
```

```
0.18923467767545252
```

IV. 结果

模型的评价与验证

基于 Xgboost 原生接口的分类，设定进行 2500 次迭代，输入提取的特征数据，训练模型。

```
d_train = xgb.DMatrix(X_train, label=y_train)
d_valid = xgb.DMatrix(X_valid, label=y_valid)
watchlist = [(d_train, 'train'), (d_valid, 'valid')]
bst = xgb.train(params, d_train, 2500, watchlist, early_stopping_rounds=50, verbose_eval=50)
```

```
[1500]  train-logloss:0.154475  valid-logloss:0.183579
[1550]  train-logloss:0.15355   valid-logloss:0.183482
[1600]  train-logloss:0.152603  valid-logloss:0.183419
[1650]  train-logloss:0.151692  valid-logloss:0.183368
[1700]  train-logloss:0.150738  valid-logloss:0.183301
[1750]  train-logloss:0.149782  valid-logloss:0.183249
[1800]  train-logloss:0.148893  valid-logloss:0.183178
[1850]  train-logloss:0.148029  valid-logloss:0.183129
[1900]  train-logloss:0.147167  valid-logloss:0.183064
[1950]  train-logloss:0.146335  valid-logloss:0.182988
[2000]  train-logloss:0.145458  valid-logloss:0.182956
[2050]  train-logloss:0.144658  valid-logloss:0.182937
[2100]  train-logloss:0.14385   valid-logloss:0.182937
Stopping. Best iteration:
[2067]  train-logloss:0.144369  valid-logloss:0.182917

0.18294025406641415
```

最后训练得到最优的验证集 logloss 值 0.183，效果较好。对测试集进行预测。

```
d_test = xgb.DMatrix(X_test)
p_test = bst.predict(d_test)
sub = pd.DataFrame()
sub['test_id'] = test['test_id']
sub['is_duplicate'] = p_test
```

kaggle 提交预测结果，得分如下：

Submission and Description	Private Score	Public Score
submission.csv 8 days ago by Gene	0.16191	0.15885

Private Socre 和 Public Socre 得分都要优于基准得分 0.463。在 public Leaderboard 排名为 522，在 private Leaderboard 排名为 486，符合项目要求的条件（提交分数需要达到 kaggle private leaderboard 的 top 20%,对于该题目的就是 660th/3307,对应 logloss 得分为 0.18267）。

V. 项目结论

结果可视化

特征工程是本项目最重要、最核心的工作，图 1 对输入模型特征的重要性进行了排序，从图中可以看出，TF-IDF Features、Word2vec features 的作用较为突出，在排序中大部分都处在较前的位置，尤其是 tfidf_wm 特征的重要性评分最高。另外，非 nlp 特征的 leaky features 也发挥了较明显的作用。

对项目的思考

本文研究了如何检测句子相似度问题。整个过程显示，Xgboost 是一种非常有效的算法，同时特征工程是本项目极其核心的工作，尤其 TF-IDF Features、Word2vec features 的作用明显，针对这一部分，我们可以进一步进行挖掘。例如，我们使用了 GloVe 通用嵌入，我们的结果与任何一般的问答系统都更相关。而如果使用 Quora 自己的 Quora 语料数据库中的单词嵌入，应该更适合 Quora 的问题格式。另一方面，数据集句子对中的语句都是单独以问题的形式出现的，如果知道询问问题的上下文，可以适当替换某些代词，或许可以实现更高的准确性。由于我们不知道在哪个上下文环境中提出的问题，因此无法对原始数据集进行此类预处理。

需要作出的改进

主要由于受算力和时间，尤其是硬件的限制，更多有价值、有意义的工作无法深入展开。应该更好地管理时间，进行深入数据探索，挖掘更多的特征，尝试多种模型的构建和堆叠，尤其是应实践检验深度模型。扩大建模管道规模，以便在不同的超参数、数据子集上构建更多模型。最近几年自然语言处理领域取得了飞速的发展，基本几项重要的成果都集中在了预训练上，包括 18 年火爆一时的 BERT，今年上半年发布的 XLNet，近期发布的 RoBERTa 都是以更好的预训练为目的的。

谷歌Lab近日发布了一个新的预训练模型"ALBERT"全面在SQuAD 2.0、GLUE、RACE 等任务上超越了 BERT、XLNet、RoBERTa 再次刷新了排行榜！ALBERT 是一种轻量版本的 BERT，利用更好的参数来训练模型，但是效果却反而得到了很大提升。ALBERT 的核心思想是采用了两种减少模型参数的方法，比 BERT 占用的内存空间小很多，同时极大提升了训练速度，更重要的是效果上也有很大的提升。这将是下一步改进的重点。

参考资料：

- 1、https://github.com/howardyclo/Kaggle-Quora-Question-Pairs
- 2、https://www.kaggle.com/sudalairajkumar/simple-leaky-exploration-notebook-quora?scriptVersionId=1184830
- 3、https://www.kaggle.com/c/quora-question-pairs/discussion/31179
- 4、https://www.cnblogs.com/wj-1314/p/9402324.html
- 5、https://www.kaggle.com/davidthaler/how-many-1-s-are-in-the-public-lb
- 6、https://www.kaggle.com/act444/lb-0-158-xgb-handcrafted-leaky
- 7、https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur/
- 8、https://www.zhihu.com/question/41354392
- 9、https://xgboost.readthedocs.io/en/latest/tutorials/model.html
- 10、https://www.c nblogs.com/viredery/p/document_similarity.html

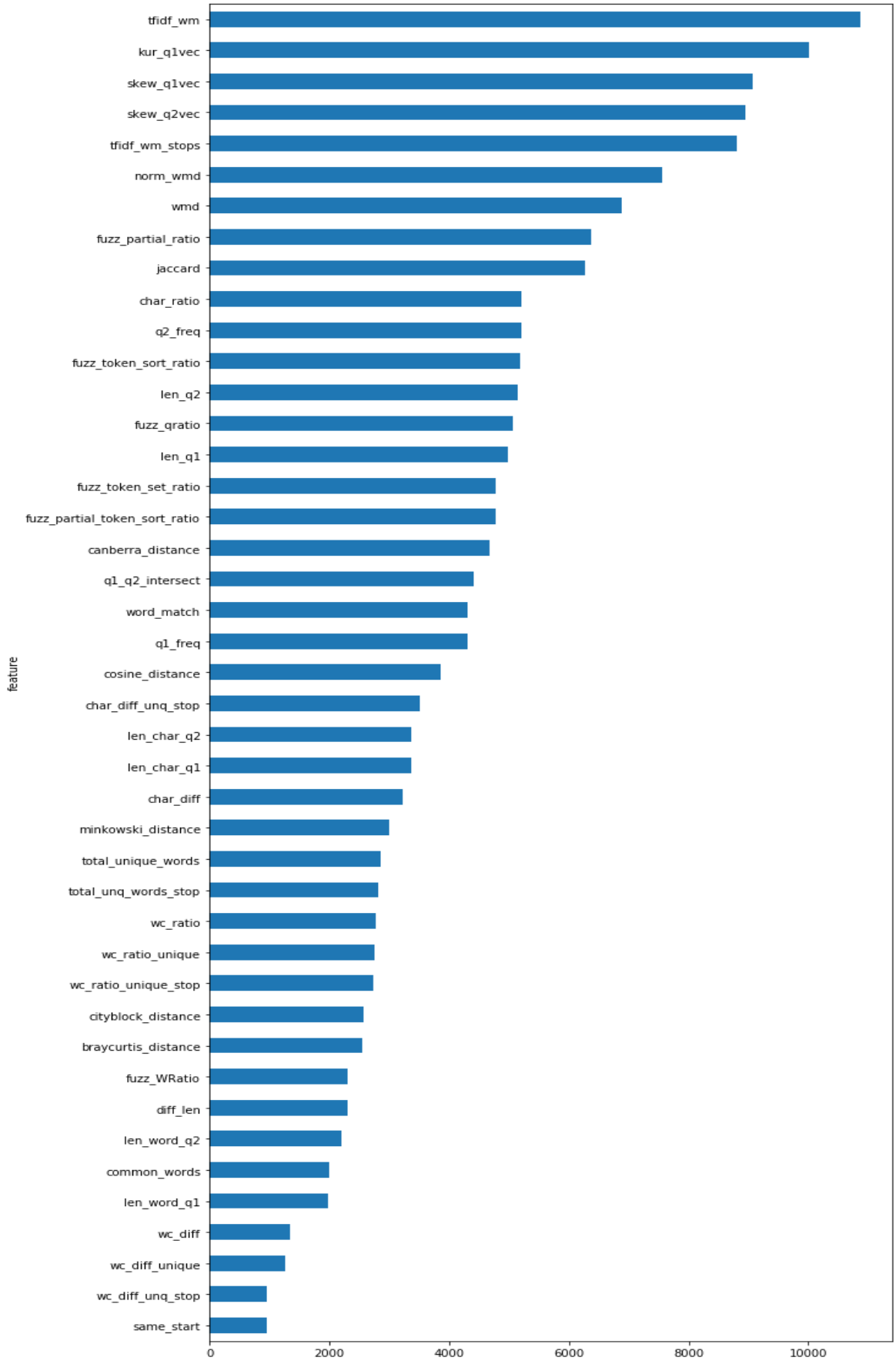

tfidf_wm
kur_q1vec
skew_q1vec
skew_q2vec
tfidf_wm_stops
norm_wmd
wmd
fuzz_partial_ratio
jaccard
char_ratio
q2_freq
fuzz_token_sort_ratio
len_q2
fuzz_qratio
len_q1
fuzz_token_set_ratio
fuzz_partial_token_sort_ratio
canberra_distance
q1_q2_intersect
word_match
q1_freq
cosine_distance
char_diff_unq_stop
len_char_q2
len_char_q1
char_diff
minkowski_distance
total_unique_words
total_unq_words_stop
wc_ratio
wc_ratio_unique
wc_ratio_unique_stop
cityblock_distance
braycurtis_distance
fuzz_WRatio
diff_len
len_word_q2
common_words
len_word_q1
wc_diff
wc_diff_unique
wc_diff_unq_stop
same_start
feature
0
2000
4000
6000
8000
10000

图 1　特征重要性排序