Table of Contents

句子相似度匹配项目 *2*

问题定义 **2**

项目概述 2

问题陈述 2

评价指标 4

分析 **4**

数据的探索 4

句子长度特征可视化分析 5

WMD 特征可视化分析 8

相似度特征可视化分析 10

算法和技术 **13**

变量说明 13

使用模型 16

基准模型 17

方法 **18**

数据预处理 18

特征提取过程 19

执行过程 22

结果 **24**

模型的评价与验证 24

合理性分析 25

项目结论 **25**

结果可视化 25

对项目的思考 26

需要作出的改进 26

引用 **27**

附录 **28**

句子相似度匹配项目

问题定义

项目概述

Quora Question Pairs 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，从而来判断句子是否一致。

该问题来自于 Quora 问答网站，在该网站上，用户可以在上面提问，并有其他用户给出高质量的回答或是独特的想法。这能鼓励人们互相学习，了解更多的知识。

每个月，超过一亿的用户会访问 Quora。在这么大的访问量下，经常会有用户提出相似的问题。相似的问题会导致用户花费更多时间去寻找最佳答案，来回答他们想问的问题。而且也会让回答问题的用户觉得，对于同一个问题，他们需要回答多次。因此在 Quora 上，一个问题的最典型形式是很有价值的。它能给提问者和回答这提供最好的用户体验。

目前，Quora 使用的是一个随机森林模型来寻找类似的问题。而本次项目就是要探索使用更先进的技术来判断给定的两个问题是否是重复问题。高准确率的判断能够帮助网站找到每个问题最佳的回答，从而提高提问者，回答者以及网站浏览者的用户体验。这就是该项目的出发点。

而在技术层面上，该问题属于二分类问题。给定数据后，我们需要训练模型要将数据划分到“相同问题”和“不同问题”两个类别中。同时，给定的训练数据中包含了人工分类后的标签结果，所以该分类任务属于监督学习中的问题。最后，由于我们处理的数据属于自然语言，所以该问题也涉及到自然语言处理。

Quora 数据集训练集共包含 40K 的句子对，且其完全来自于 Quora 网站自身。句子对中每个句子以字符串的形式存储，每个字符串即句子的自然语言表达。

问题陈述

这个项目中要解决的问题是判断两个问题是否表达相同的意思。

使用的训练数据集主要包含三列数据：第一列为问题 1，数据类型为字符串吗，包含了问题对中第一个问题的自然语言表示形式，使用语言为英语。第二列数据为问题 2，数据类型和内容与第一列数据类似，包含的是问题对中第二个问题的自然语言表达。第三列是表示问题对中两个问题是否相同的一个标签，该标签为数值类型，只包含 0,1 两个值。0 代表两个问题意思不同，1 代表两个问题意思相同。数据样例如下：

	question1	question2	is_duplicate
0	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0
5	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...	1
6	Should I buy tiago?	What keeps childern active and far from phone ...	0
7	How can I be a good geologist?	What should I do to be a great geologist?	1

解决方案

这个问题是分类问题，所以使用分类模型来解决该问题。方案的基本思路按照分类模型的训练过程进行：首先从原始数据中提取特征，然后选择并训练模型，并进行模型调参。最后，选择效果最好的模型，并用该模型解决问题。具体细节如下

1. 特征提取
   (1) 原始数据为自然语言，无法直接作为分类模型的输入，因此要在自然语言的基础上生成一些特征，例如字符串的长度，包含单词数量等。
   (2) 由于原始数据为自然语言，因此可以使用自然语言处理的技术来生成特征。例如使用词袋模型，tf-idf 模型，句子模糊匹配等技术对自然语言处理，并生成与自然语言相关的特征。
   (3) 主题模型特征。词袋模型和 tf-idf 模型将语句转化成维度很高的向量。在本项目中该向量约有 3 万个维度。这样的数据不便处理，且包含的信息中包含很多噪音，分类效果不理想。因此，使用主题提取技术处理数据。提取出语句中包含的主题信息，同时也起到降维的作用。得到主题模型后，每个语句将被转化成维度较少的向量。然后可以使用得到的向量计算两个语句的相似度特征。同时，降维后的数据可以直接作为特征数据。
2. 模型训练

(1) 数据清洗：通过特征提取获得了特征数据，但该数据中往往包含一些异常数据。例如空值，或数据范围过大的数据。因此在训练模型前需要对数据进行清理
(2) 分割训练数据集和测试数据集。由于后续涉及到模型选择，还需要检验数据集（validation data set）
(3) 对选取的模型进行调参

3. 模型选择
(1) 模型训练过程中会选取多个分类模型进行训练。在最后比较模型间性能差异，通过测试数据集选择最好的模型来解决问题

以上为解决问题的过程。最终期望的结果是能够获得效果明显的模型。通过该模型判断给定的两个句子是否含有相同得到意思。

评价指标

logloss 是 kaggle 指定的评估标准，所以最终决定使用这个评价标准。

Logloss, 即对数似然损失(Log-likelihood Loss), 也称逻辑斯谛回归损失(Logistic Loss), 是在概率估计上定义的.它常用于(multi-nominal, 多项)逻辑斯谛回归和神经网络,以及一些期望最大化算法的变体. 可用于评估分类器的概率输出.

对数损失通过惩罚错误的分类,实现对分类器的准确度(Accuracy)的量化. 最小化对数损失基本等价于最大化分类器的准确度.为了计算对数损失, 分类器必须提供对输入的所属的每个类别的概率值, 不只是最可能的类别. 对数损失函数的计算公式如下:

$$L(Y,P(Y|X))=-\log P(Y|X)=-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}y_{ij}\log(p_{ij})$$

其中, Y 为输出变量, X 为输入变量, L 为损失函数. N 为输入样本量, M 为可能的类别数, y_{ij} 是一个二值指标, 表示类别 j 是否是输入实例 x_i 的真实类别。p_{ij} 为模型或分类器预测输入实例 x_i 属于类别 j 的概率。Logloss 越小表示其预测效果越好。

在实际模型训练过程中，我们将会使用 sklearn 中的 train_test_split 工具对训练数据集进行分割，以获取训练数据集和测试数据集。并用我们选取的指标，准确率， f1 分数和 logloss，测试模型在训练和测试数据集上的性能表现，以此作为评估标准。

分析

数据的探索

使用的数据为 Quora Question Pairs, 该数据集中使用的训练数据集包含三个列：question1， question2， is_duplicate。这三个列分别代表了句子对中的第一个句子，第二个句子和句子对是否表示相同意思的标签。该标签为数值类型，只

包含 0 和 1 两个数值。其中 0 表示句子对意思不同，1 表示句子对意思相同。为了方便后面的分析，在后文中，我将意思相同的数据成为正样本，意思不同的数据称为负样本，而 is_duplicate 字段可以被称为我们的目标字段

数据集基本参数如下：

数据量	404290	列数	3
句子最大长度	1169	句子最小长度	1
包含单词最大值	237	包含单词最小值	1

表 1

句子的长度和单词数量在一定程度上反映了句子包含的语义信息的量，而相同含义的句子在大部分情况下会含有相同或相近的单词数量或长度。所以可以使用句子长度和单词数量作为判断句子对意思是否相同的特征。

句子长度特征可视化分析

句子对中的句子长度和包含的词数等信息能够用来判断句子对是否能够表示相同含义。一般而言，在提问时，用户往往会用相似的词语和句子结构来描述相同的问题，而相似的句子结构和词语会使句子长度也比较相似。因此，在直觉上来说，相似长度的句子是相似句子的可能性更大。因此我们会分析句子长度相关的统计量。具体如下：

句子长度：句子字符串的长度

句子字符长度：句子中去除空格等空白符号后的字符长度

句子单词数量：用空格分割句子后得到的列表的长度。该长度为句子中单词和特殊符号的数量和。

进一步数据处理：直接获取到的长度特征为一个二元组，分别包含了第一个和第二个句子的长度特征。这个二元组不方便可视化，所以我将二元组的两个数据做比值并进一步处理来获取最终方便可视化的数据。以句子长度为例，处理过程如下：

句子长度的原始数据为二元组：（len_q1, len_q2）

将数据做比值，获得一个数据值：len_q_ratio = len_q1 / len_q2

为了反映该比值对判断的作用，我根据该比值将数据划分为较小的区间，并计算每个小区间中的数据中 is_duplicate 段的平均值，平均值越大的区间表示该区间包含越多的正样本，而平均值越小，表示该区间包含的正样本越少。

但在可视化之前，发现 len_q_ratio 比值的分布空间为[0.006711409395973154, 117.0]。因为是比值，数据应该以 1 为中心分布，因此数据分布不均匀，不利于可视化，所以考虑使用对数操作调整数据分布。

比值对数：len_q_ratio_ln = ln(len_q1 / len_q2)

该数值的分布大约在[-4，4]上。以 0.04 为小区间的划分范围，获得最终可视化分布如下：

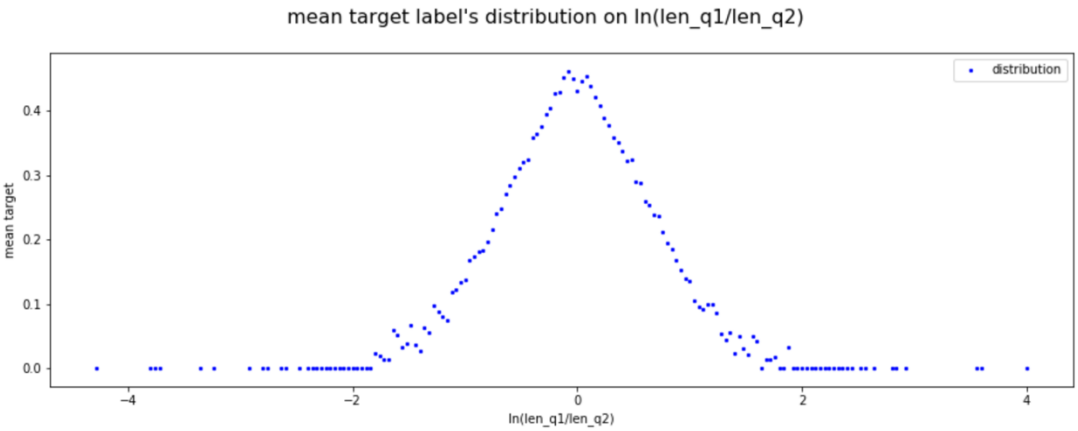

图 2-2-1

上图中和坐标为 ln(len_q1/len_q2)，是我最终选定的比值对数特征。纵坐标为每个小区间中数据 is_duplicate 字段的均值。

从图中可见，当比值对数在[-4, -2]和[2,4]之间时目标字段均值为 0，说明该句子对长度差异过大时，基本不可能是相同含义的句子。而比值对数越接近 0，目标字段均值越大，说明句子长度越相近，句子对的含义越肯能相同。因此该特征值对于模型的预测越有效果。

同时，我也对该比值的分布进行了可视化：

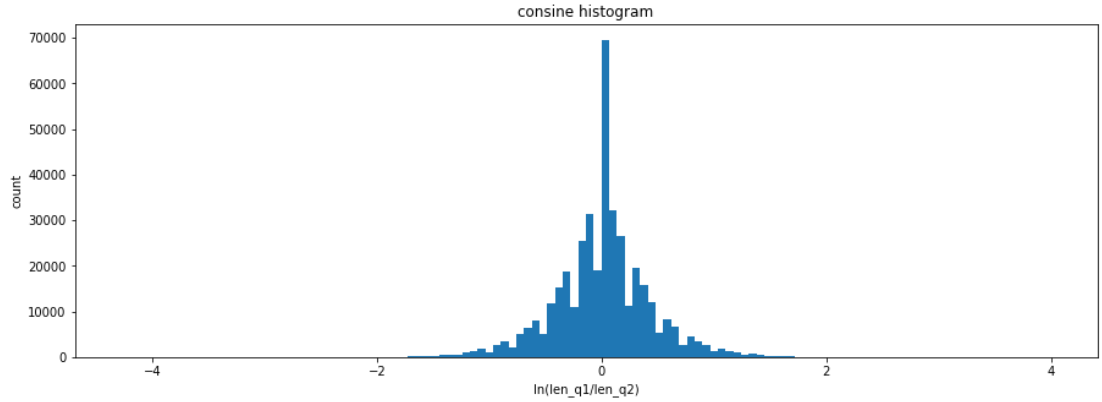

图 2-2-2

同上，我也准备了句子字符长度和句子单词数量对应的比值对数分布，如下：

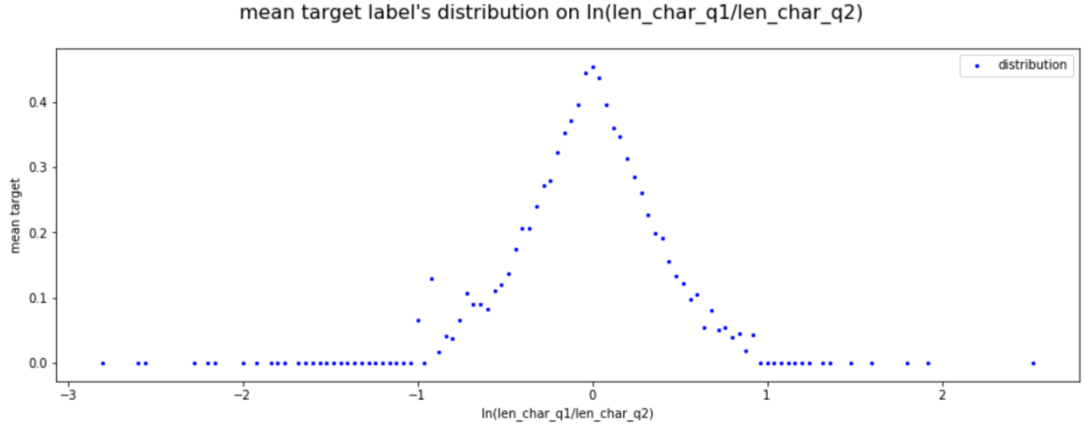

图 2-2-3

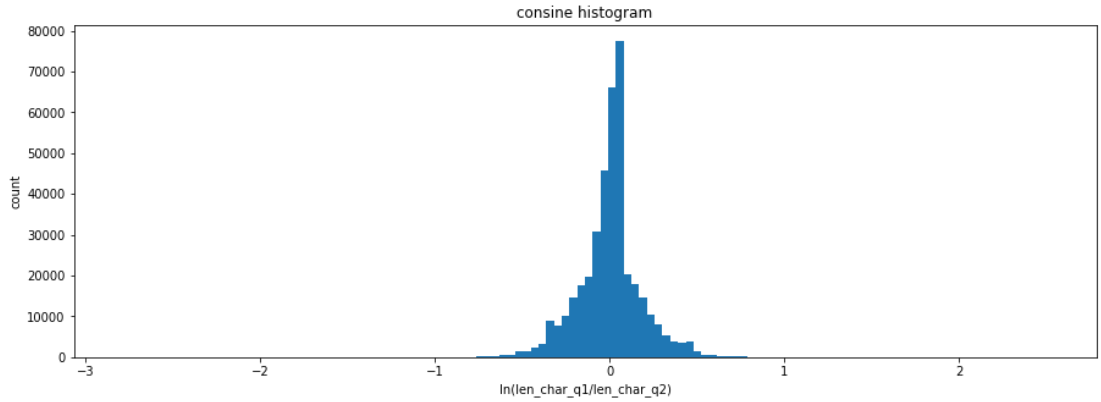

图 2-2-4

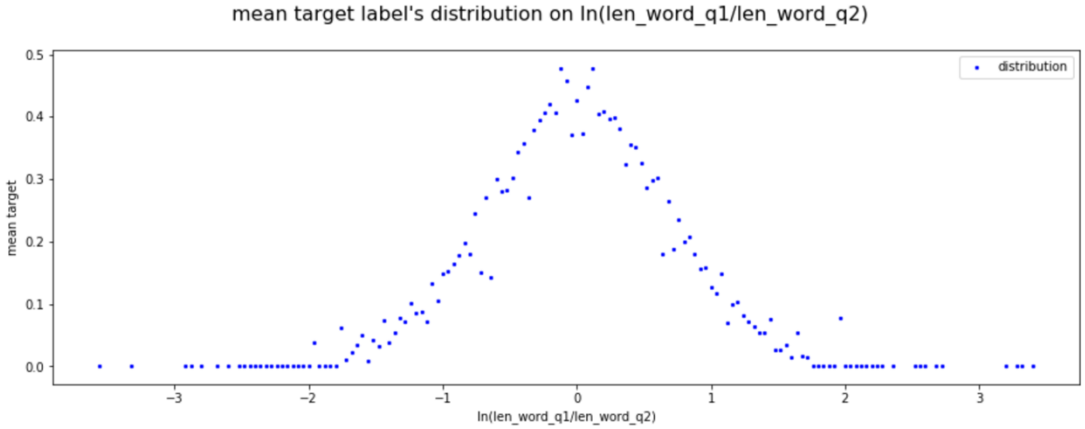

图 2-2-5

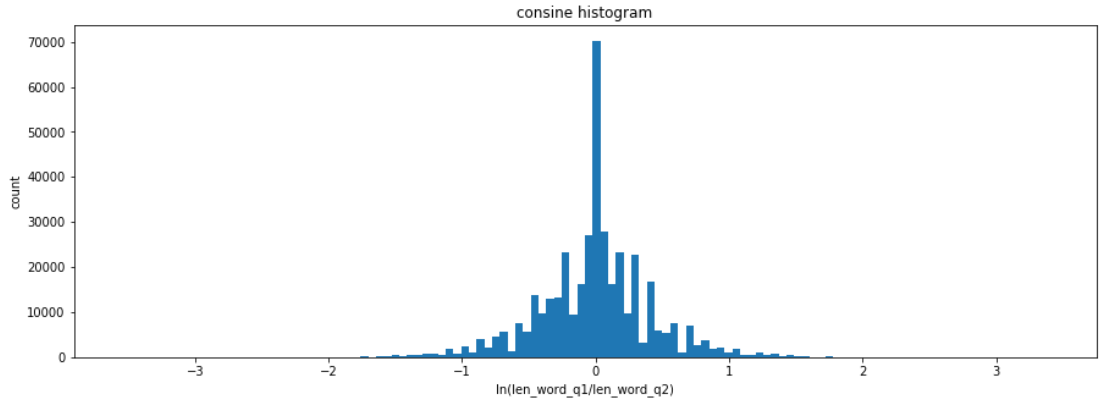

图 2-2-6

发现分布与字符串长度的分布类似。

在此之后，再分析一下句子对中同时出现的单词数量与目标字段的关系。其中相同单词比例可定义为：

Commen = 2 * commen_count / (len1 + len2)

其中 commen_count 为两个句子中同时出现的单词的数量，len1 和 len2 分别为句子 1 和句子 2 包含的单词数量。

然后显示目标字段的均值在该比例上的分布，如下图：

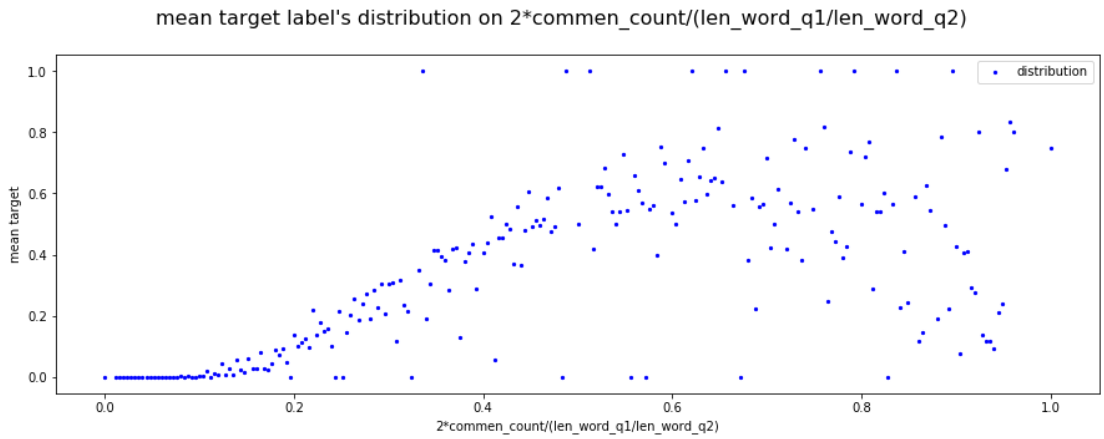

图 2-2-7

从图 2-2-7 中可以看到，在比值较小时目标字段的均值较低，随着比值的升高，目标字段的均值分布开始变高。可见该比值对于句子对判断有作用。

而该比值的分布如下图：

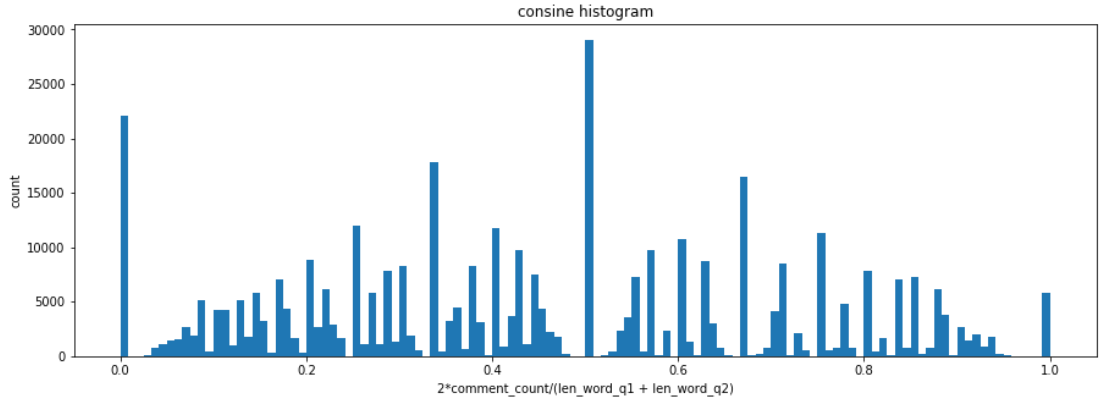

图 2-2-8

从图 2-2-8 中可以看到，数据分布相对均匀，结合图 2-2-7，可以发现，该比值能够帮助我们对大量数据进行划分。

WMD 特征可视化分析

WMD 词移距离（WMD）是基于 word2vec 用来表示两个文本间语义距离的技术。其中 Word2Vec 技术将词映射为一个词向量，在这个向量空间中，语义相似的词之间距离会比较小。

Word2Vec 得到的词向量可以反映词与词之间的语义差别，那么如果我们希望有一个距离能够反映文档和文档之间的相似度，可以将文档距离建模成两个文档中词的语义距离的一个组合，比如说对两个文档中的任意两个词所对应的词向量求欧氏距离然后再加权求和，形式如下：

$$\sum_{i,j=1}^{n} T_{ij}c(i,j)$$

其中 c(i,j)为 i,j 两个词所对应的词向量的欧氏距离。而其中的加权矩阵 T 会保证由文档 1 中的某个词 i 移动到文档 2 中的各个词的权重之和应该与文档 1 中的这个词 i 的权重相等。同样，文档 2 中的某个词 j 所接受到由文档 1 中的各个词所

流入的权重之和应该等于词 j 在文档 2 中的权重。加权代价求和后在经过线性规划求得下界之后，即可作为文档 a 中单词转移到文档 b 中单词的最短总距离，代表两个文档之间的相似度。

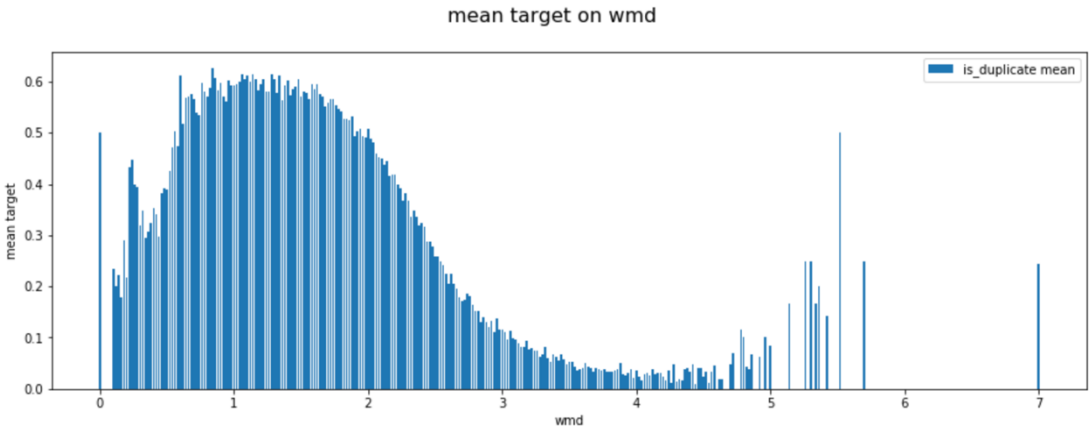

图 2-3-1

从可视化结果中可以看到在[0, 2.5]区间中，目标字段的取值较高，包含的正样本较多，而在[2.5, 5.5]区间中，取值较低，包含正样本数量较少。在[5, 6]区间中有一些目标字段均值较高的区间，但对比了 wmd 直方图，发现该区间内分布的数据量很少，所以可以忽略。

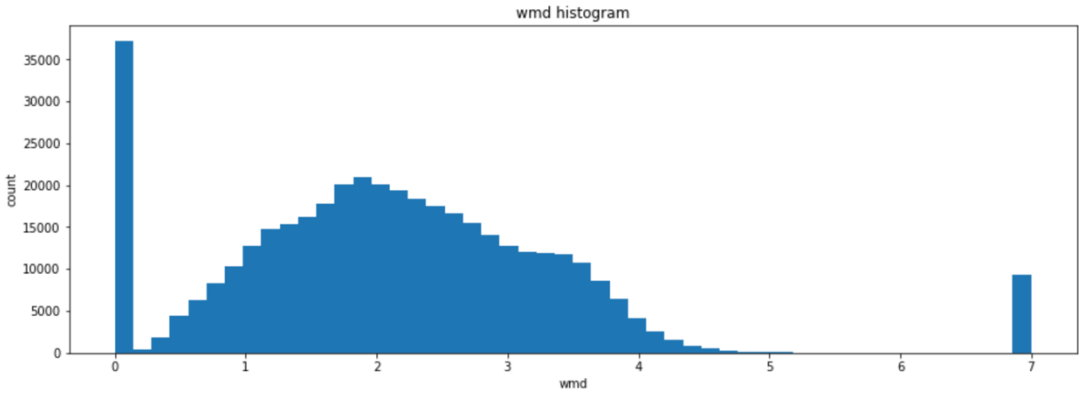

图 2-3-2

同样，对于 Normal WMD，使用同样的可视化方法得到如下可视化图片。同样可以看到，以 1.0 为划分点，可以将数据划分为目标字段均值较高和均值较低的区间。可见，Normal WMD 同样对数据有划分作用。

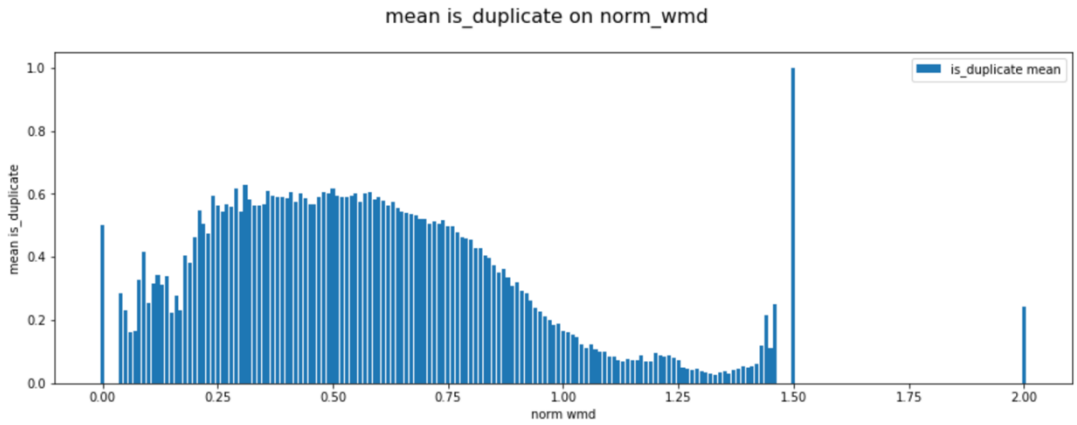

图 2-3-3

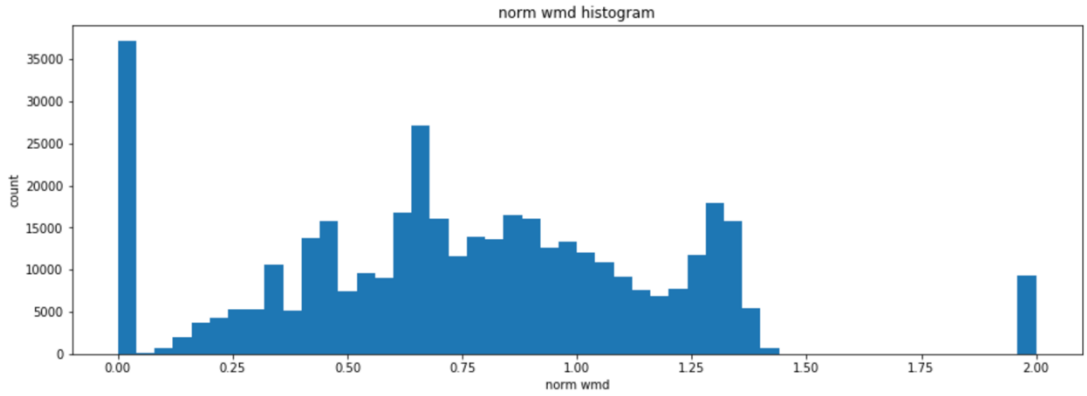

图 2-3-4

相似度特征可视化分析

项目中使用了距离相关的特征来描述句子对之间的差异。在文档中将可视化较为典型的两个距离分布。

以余弦相似度为例，距离相关特征的计算方法如下:

以句子对包含的两个句子为语料库，计算句子中每个单词的 tf-idf 值，并以单词对应的 tf-idf 值将两个句子转化为两个向量。获取向量后计算两个向量的余弦相似度。

余弦相似性通过测量两个向量的夹角的余弦值来度量它们之间的相似性。0 度角的余弦值是 1，而其他任何角度的余弦值都不大于 1；并且其最小值是-1。

Cityblock distiance 又被称为曼哈顿距离，是向量中各个维度数值查的绝对值的和。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

$$|x_1 - x_2| + |y_1 - y_2|.$$

Jaccard distance（雅卡尔距离）用于量度样本集之间的不相似度，其定义为 1 减去雅卡尔系数。而雅卡尔指数是用于比较样本集的相似性与多样性的统计量。雅卡尔系数能够量度有限样本集合的相似度，其定义为两个集合交集大小与并集大小之间的比例：

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

因而雅卡尔距离的定义为：

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

Canberra Distance(堪培拉距离)被用来衡量向量空间中两个点之间的距离，它是曼哈顿距离的加权版本。其定义公式为：

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

其中 p,q 为作比较的两个向量。通常 Canberra distance 对于接近于 0（大于等于 0）的值的变化非常敏感。

Euclidean distance 既几何距离，其计算公式如下：

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Bray-Curtis distance 与曼哈顿距离相似，其计算公式为：

$$d_{ij} = \frac{\sum_{k=1}^{n} |x_{ik} - x_{jk}|}{\sum_{k=1}^{n} (x_{ik} + x_{jk})}$$

与曼哈顿距离相比，该距离的特点是：当比较的两个向量的所有维度数据都是正值时，距离的结果始终在[0, 1)之间，这样不用再做数据的归一化。

鉴于选用的距离特征较多，且实际可视化过程中发现大部分数据的分布有相似性，在此只展示余弦相似度和曼哈顿距离的可视化结果，如下图。

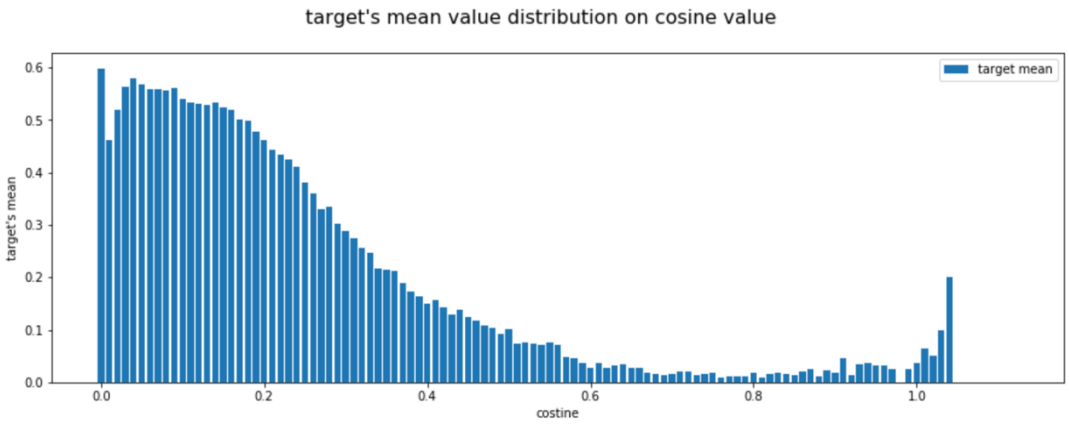

图 2-4-1

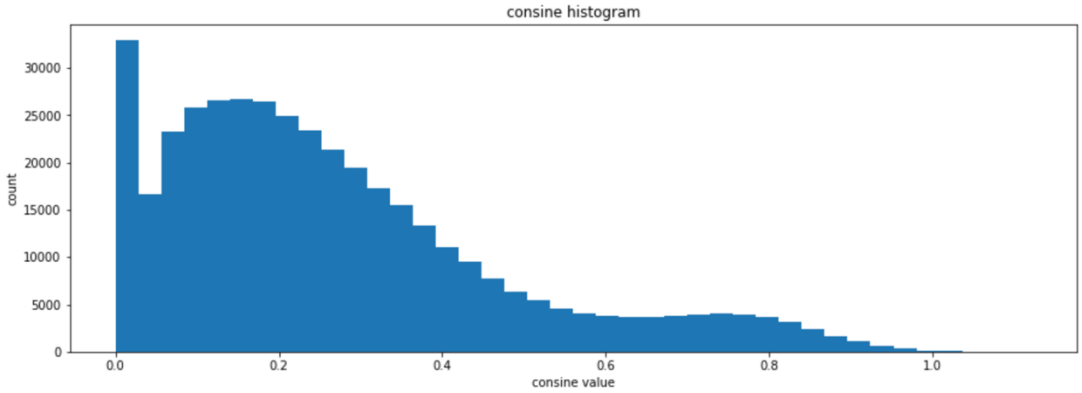

图 2-4-2

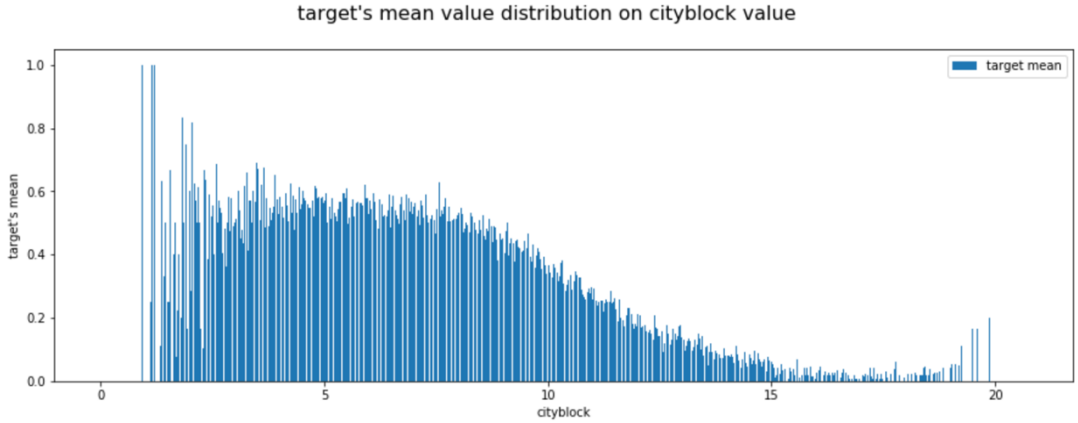

图 2-4-3

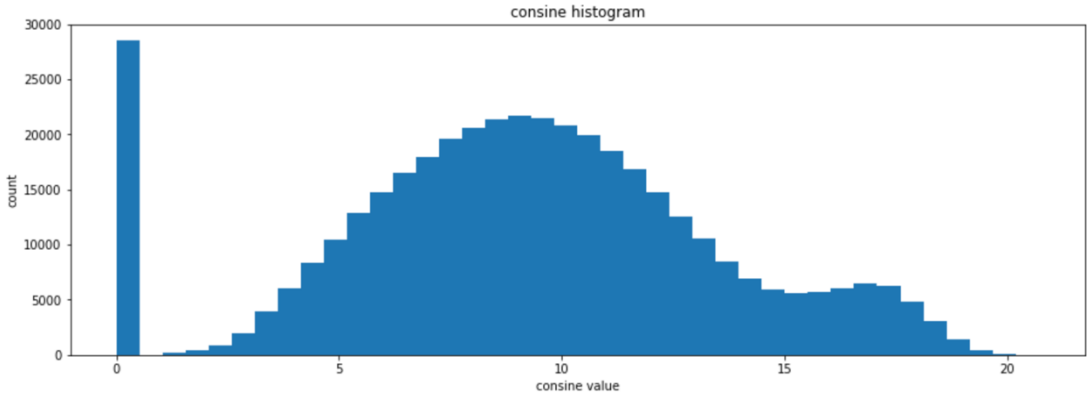

图 2-4-4

图 2-4-1 和 2-4-2 分别是目标字段的均值在余弦相似度和曼哈顿距离上的分布。两者分布大致类似，在分布区间的前半部分，目标字段均值较高，后半部分均值较低。可见特征对于分类还是有帮助的。

而通过图 2-4-3 和 2-4-4，可以看到余弦相似度和曼哈顿距离本身的分布情况。

同时发现，余弦相似度中，相似度数值较高（大于 0.4）的部分中，数据分布稀少。而目标字段均值的区分点也在 0.4 左右。这说明通过余弦相似度能够找出的负样本的数量过少，根据此距离进行分类的效果肯能不会太好。而图 11 中发现，通过曼哈顿距离分布的前半部分和后半部分的数据量相近，这样通过曼哈顿距离区分出的负样本数量较多，分类效果要更好。

算法和技术

变量说明

使用的变量是我们从原始的句子对数据中根据自然语言处理技术，获取到的变量，包括如下几方面的变量：

一， 句子长度相关变量
二， WMD 特征变量
三， 模糊匹配相关变量
四， 相似度相关变量
五， 主题模型相关变量

句子长度相关变量

本次项目使用的长度相关变量包括：len_q1, len_q2, diff_len, len_char_q1, len_char_q2, len_word_q1, len_word_q2, common_words 等。在数据分析部分中，我们已经看到句子对的长度和长度差异相关的变量与句子是否表示相同含义有关，可以用来作为判断依据。

WMD 相关变量

本次项目中包括 wmd, norm_wmd 两个 WMD 相关变量。该变量是根据词移距离计算两个句子的语义距离的变量，可以直接用来进行训练。

模糊匹配相关变量

本次项目中包含如下模糊匹配相关变量：fuzz_qratio, fuzz_WRatio, fuzz_partial_ratio, fuzz_partial_token_set_ratio, fuzz_partial_token_sort_ratio, fuzz_token_set_ratio, fuzz_token_sort_ratio, wmd, norm_wmd, cosine_distance, cityblock_distance, jaccard_distance, canberra_distance, euclidean_distance, minkowski_distance, braycurtis_distance, skew_q1vec, skew_q2vec, kur_q1vec, kur_q2vec。这些变量是根据句子中包含的单词间的编辑距离，计算出的字符串之间的相似度。而从直觉上来看，句子的文本相似度越高，其语义也就越相似，因此可以用来作为训练数据。

相似度相关变量

本次项目使用的相似度包括： cosine_distance, cityblock_distance,

jaccard_distance, canberra_distance, euclidean_distance, minkowski_distance, braycurtis_distance。

这些相似度计算的输入都是向量，而为了得到相似度相关变量，我们还需将句子对提取成两个向量。而本次试验中使用的向量化技术包括如下几种：

word2vec

顾名思义，Word2Vec 就是把单词转换成向量。它本质上是一种单词聚类的方法，是实现单词语义推测、句子情感分析等目的一种手段。

其实现可以认为是逻辑回归网络的一种变形。以 CBOW 为例，还是每次挨个把语料库的词取出来，作为该次训练的目标，然后把这个词所在位置的前后 N 个词（N 通常用 1 或者 2，数字越大学习到的模型信息量越丰富，但需要的训练时间越长）依次作为训练的输入。比如 N 值为 2，每取出一个词（作为中心词），就要使用该词的前后各 2 个词（作为环境词），分别放到网络训练一次，一共要取 4 个环境词。还是以识别 5 万个词的向量为例，具体训练过程如下：

首先预处理数据，把所有需要进行训练的词汇编上序号，比如 1-50000

随机初始化一个维度为 50000x50 的矩阵，作为待训练的嵌入矩阵

每次取出一个中心词和它的其中一个环境词

以环境词编号作行数，从词向量矩阵里取出这一行数据（50 维向量）

将这个 50 维向量作为逻辑回归网络的输入，训练目标是中心词编号相应的 One-Hot 向量

在训练的反向传播时计算，不但更新逻辑回归网络的权重矩阵，还要往前多传递一级，把取出的 50 维向量的值也根据目标梯度进行更新

将更新过的 50 维向量重新更新到嵌入矩阵相应的行

重复以上过程，直到所有的中心词都已经被遍历一遍，此时嵌入矩阵值的计算就完成了

整个训练过程就是直接对这个矩阵进行更新。得到的矩阵的每一行就对应着以 One hot 编码形式对应着的单词。在这个例子中就将 5 万个单词转化为 5 万个 50 维的向量。

本次项目中使用的 word2vec 包含两部分：使用通用 word2vec 模型生成的向量，使用数据集本身作为语料库，训练模型生成的变量。

通用 word2vec 模型结果

通用 word2vec 模型使用的是 Google News vectors negative300，这是 Google 训练好的用来进行 word2vec 的模型。训练这个模型的语料库规模很大，因此该模型效果很好。

基于预测数据产生的 word2vec 模型结果

而使用训练数据集生成的向量则是用 gensim 提供的 word2vec 的库，将数据集中包含的句子作为输入，训练出的模型。由于使用的语料库就是要预测的数据集本身，其效果会比较好。

在获取到 word2vec 向量后，可以根据句子中包含的单词，将句子转化成一个向量。而之后就可以生成相似度相关的变量。

tf-idf

TF-IDF 是一种用于信息检索与文本挖掘的常用加权技术，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。tf-idf 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。

但由于单词数量很多，tf-idf 转化后的向量维度极大，在本项目中转化后每个句子有 3 万以上的维度。而由于每个句子包含的词的数量相对较少，因此向量中大部分数据为 0。这样的数据维度过大，而且其中 0 值很多，不便于直接计算。后面会使用主题模型为向量降维。

主题模型

在机器学习和自然语言处理等领域是用来在一系列文档中发现抽象主题的一种统计模型。直观来讲，如果一篇文章有一个中心思想，那么一些特定词语会更频繁的出现。比方说，如果一篇文章是在讲狗的，那“狗”和“骨头”等词出现的频率会高些。如果一篇文章是在讲猫的，那“猫”和“鱼”等词出现的频率会高些。而有些词例如“这个”、“和”大概在两篇文章中出现的频率会大致相等。但真实的情况是，一篇文章通常包含多种主题，而且每个主题所占比例各不相同。因此，如果一篇文章 10%和猫有关，90%和狗有关，那么和狗相关的关键字出现的次数大概会是和猫相关的关键字出现次数的 9 倍。一个主题模型试图用数学框架来体现文档的这种特点。主题模型自动分析每个文档，统计文档内的词语，根据统计的信息来断定当前文档含有哪些主题，以及每个主题所占的比例各为多少。通常使用的主题模型包括：SVD，NMF，LSI，LDA 等。

奇异值分解(Singular Value Decomposition，以下简称 SVD)是在机器学习领域广泛应用的算法，它不光可以用于降维算法中的特征分解，还可以用于推荐系统，以及自然语言处理等领域。该算法通过对矩阵进行奇异值分解，获取矩阵的特征值。并通过保留绝对值较大的特征值，舍弃较小特征值的方法，结合一定数学方法，达到数据降维的效果。

LDA 也是一种主题模型，它可以将文档集 中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题（分布）出来后，便可以根据主题（分布）进行主题聚类或文本分类。同时，它是一种典型的词袋模型，即一篇文档是由一组词构成，词与词之间没有先后顺序的关系。

NMF 与 SVD 相似，但该方法中保证矩阵中均未非负数，且该算法相对于 SVD，计算方式更优化，速度更快。

主题模型的使用可以将句子的字符串转化为长度适当的向量。转化后可以使用相似度算法计算出各种相似度特征。

其次，主题模型本身也是自然语言的一种高度抽象，能够反映句子中描述的主题，因此，主题模型生成的向量本身也可作为特征。

词性数据

词性数据指的是句子中包含各种词性的词的信息。

句子文本是由一个个单词组成，每个单词具有词性，表示该单词是名词，动词等等。语义相近的句子，其结构应该相近，句子中不同词性的单词数量就会比较相近。因此这些词的词性统计数据就会相近。这样以各种词性单词的统计数量组成的向量，其相似距离就会很小。

图相关变量

如果把句子间相似看作一种关系，那么句子就是一个实体。这样就可将句子数据转化成一个图结构，其中句子看作节点，两个句子若相似就可以在对应的节点间连接一条线。这样就得到一个图结构的数据。同时，由于相似这个概念本身具有传递性。例如，句子 A 若和句子 B 相似，句子 B 和句子 C 相似。那么句子 A 就很可能和句子 C 相似。

按照这样的传递关系，可以将所有相似的句子划归到同一个节点群中。当要判断新的句子对是否相似时，可以查看这两个句子是否属于同一个节点群。

这种方法完全没有考虑句子本身自然语言相关的特性，因此可以作为自然语言处理方法的一种补充。相当于，当自然语言处理无法直接将两个相似的句子判断出来时，可以用之前判断的一些句子信息，将这两个句子相似与否的结果推断出来。

使用模型

解决的问题为一个典型的分类问题，所以可以使用分类或回归模型。

该项目中我尝试使用了 RandomForest, GBDT, SGD, LogisticRegression 以及 lightGBM 算法。最终选择了 lightGBM 算法。

GBDT(Gradient Boosting Decision Tree) 又叫 MART（Multiple Additive Regression Tree)，是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的结论累加起来做最终答案。它在被提出之初就和 SVM 一起被认为是泛化能力（generalization)较强的算法。我曾尝试使用 SVM，但由于训练数据量较大，SVM 训练时间非常长，最后没有尝试使用该算法。

参数说明：

在实验中使用了如下一些参数。

n_estimators: 也就是弱学习器的最大迭代次数，或者说最大的弱学习器的个数。一般来说 n_estimators 太小，容易欠拟合，n_estimators 太大，又容易过拟合，一般选择一个适中的数值。默认是 100。在实际调参的过程中，常常将 n_estimators 和 learning_rate 一起考虑。

learning_rate: 即每个弱学习器的权重缩减系数 v，也称作步长，对于同样的训练集拟合效果，较小的 v 意味着需要更多的弱学习器的迭代次数。一般来说，可以从一个小一点的 v 开始调参，默认是 1。

subsample:子采样，取值为(0,1]。GBDT 的采样是不放回抽样。如果取值为 1，则全部样本都使用，等于没有使用子采样。如果取值小于 1，则只有一部分样本会去做 GBDT 的决策树拟合。选择小于 1 的比例可以减少方差，即防止过拟合，但是会增加样本拟合的偏差，因此取值不能太低。推荐在[0.5, 0.8]之间，默认是

1.0，即不使用子采样。

决策树最大深度 max_depth: 默认可以不输入，如果不输入的话，默认值是 3。一般来说，数据少或者特征少的时候可以不管这个值。如果模型样本量多，特征也多的情况下，推荐限制这个最大深度，具体的取值取决于数据的分布。常用的可以取值 10-100 之间。

内部节点再划分所需最小样本数 min_samples_split: 这个值限制了子树继续划分的条件，如果某节点的样本数少于 min_samples_split，则不会继续再尝试选择最优特征来进行划分。 默认是 2.如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

叶子节点最少样本数 min_samples_leaf: 这个值限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。 默认是 1, 可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

LightGBM 是一个梯度 boosting 框架, 使用基于学习算法的决策树. 它是分布式的，高效的，具有以下优势: 1，速度和内存使用的优化。2，稀疏优化。3，准确率的优化。4，网络通信的优化。5，并行学习的优化。6，GPU 支持可处理大规模数据。

LightGBM 可变参数：

feature_fraction：默认值=1.0, 0.0 < feature_fraction < 1.0, 也称 sub_feature, colsample_bytree。如果 feature_fraction 小于 1.0, LightGBM 将会在每次迭代中随机选择部分特征. 例如, 如果设置为 0.8, 将会在每棵树训练之前选择 80% 的特征，该参数可以用来加速训练，也可以用来处理过拟合。

bagging_fraction：0.0 < bagging_fraction < 1.0, 也称 sub_row, subsample。类似于 feature_fraction, 但是它将在不进行重采样的情况下随机选择部分数据。可以用来加速训练，也可以用来处理过拟合

bagging_freq：也称 subsample_freq。bagging 的频率为 0 意味着禁用 bagging. k 意味着每 k 次迭代执行 bagging

lambda_l1:默认为 0,也称 reg_alpha，表示的是 L1 正则化,double 类型

lambda_l2:默认为 0,也称 reg_lambda，表示的是 L2 正则化，double 类型

cat_smooth： 默认值为 10，用于分类特征。可以降低噪声在分类特征中的影响, 尤其是对数据很少的类别

基准模型

项目中我选取了 RandomForest，GBDT，LogsitcRegression 和 SGD 四个分类模型对该问题进行预测。基准模型为这四个模型的默认参数，并给定一个固定的 random_state 以方便的进行后续的模型对比。

训练数据和测试数据生产过程如下：

Quora 给定的训练数据集中包含约 40 万条数据。通过我们之前介绍的方法生成变量后，又对数据进行了清洗。

清洗过程主要去除了数据中的空值，NaN 值以及无限大数值。最后对得到的

数据集进行去重，获取到最终可以使用的数据。

最后对数据进行切割。以 1：4 的比例分割数据，获取到训练模型时需要的验证数据集和训练数据集。

各个数据集的尺寸如下:

Train Set	Validation Set	All
39808	9952	49760

表 2

训练基本模型时，使用训练数据集新型模型训练，使用测试数据集对模型进行测试。验证数据集将会在后面的模型调优过程中使用。

因为是基准模型，使用默认值参数进行训练。得到模型的效果如下结果如下：

model	LogLoss	
	Train	Validation
RandomForest	0.13534068828676862	0.6389915118402073
GBDT	0.4463927424970769	0.46849611543040853
LogsitcRegression	0.4756306057602621	0.48807730251438775
SGD	1.52619352511403	1.698888334670201
XGBoost	0.44771961942361543	0.46771075566727116
lightGBM	0.3636642707192053	7.3164793442626905

表 3

由于最终的评价标准为 logloss，我们在选择基准模型时将以 logloss 为主要选择标准。表 3 即为几个模型未进行调参时的 logloss 预测结果。从表中可以看出，除 SGD 外，其他几个模型预测效果都相对比较好。其中效果最好的是 lightGBM 模型，训练和测试的准确率都很高。而 RandomForest 明显出现过拟合，而 SGD 则明显欠拟合。而 GBDT 与 XGBoost 效果与 lightGBM 相近，但都没有 lightGBM 下过好，考虑是没有进行对应参数调节造成的。

根据表 3 的结果，最终将默认参数的 lightGBM 模型选为基准模型。

方法

数据预处理

预处理过程主要包含三步骤：初始字符串处理，特征数据变换，特征数据清洗

一，初始字符串处理

初始字符串处理是在进行特征变量提取之前进行的。步骤如下：

1. 将字符串统一用小写字母表示
2. 将字符串以空格分割
3. 对分割后的单词进行词根提取，将不同形式的单词统一

其中第三步对单词进行词根提取时，考虑到单词往往具有不同词义，还要根据单词单词本身的词性进行提取。例如 leaves，可以是动词 leave 的第三人称单

数形式，也可以是名词 leaf 的复数形式，因此需要进行词义分析，再根据词义进行词根提取。具体代码可见 featue.py 中的 lemmatize_all（）方法。

二，特征数据变换

特征变换主要是针对分布不均的特征数据，将其进行归一化。如果两个特征的分布差异较大，例如一个分布在（0，1），另一个分布在（100， 10000）那么这些特征可能会影响某些算法的效果。需要将其调整到相同的范围，例如[0, 1]间。

但本项目中实际上没有进行特征变换，在之前的特征可视化过程中可以看到，使用到的主要特征包括：字符串长度相关特征，向量化距离特征，词移距离，模糊匹配特征。这些特征的分布范围基本都在（-10， 10）之间。模糊匹配由于使用的是百分比结果，结果分布也都在（0，100）间。我认为这个分布范围的区别不是特别大，因此不需要进行进一步的特征变换。

三，特征数据清洗

数据的异常和一些特殊值已经在数据处理过程中介绍过，主要是对 Nan，inf 等异常值进行了处理。由于其他特征分布范围不大，分布相对均匀，所以没有做额外的异常值或利群店的处理。

训练数据和测试数据生产过程如下：

Quora 给定的训练数据集中包含约 40 万条数据。通过我们之前介绍的方法生成变量后，又对数据进行了清洗。

清洗过程主要去除了数据中的空值，NaN 值以及无限大数值。最后对得到的数据集进行去重，获取到最终可以使用的数据。

最后对数据进行切割。以 1：4 的比例分割数据，获取到测试数据集和训练数据集。这两个数据集是用来进行模型训练，和模型性能比较的

特征提取过程

本次项目中主要使用了如下特征变量：句子长度特征，WMD 特征，模糊匹配特征，主题模型特征，句子间距离特征。

句子长度相关特征

句子长度特征如之前介绍的，主要是使用 python 对句子进行处理，获取字符串长度，单词数量等数据，再以这些基本长度产生长度比例等特征。具体代码请参见 feature.py 文件。

WMD 特征

WMD 特征主要是利用词移距离技术计算句子间的距离，也算是距离特征的一种。本项目中使用 gensim 库进行相关计算，具体请参见 feature.py 中 wmd_feature()和 norm_wmd_feature()方法。

模糊匹配特征

前面也已经介绍过，模糊匹配主要是根据句子中包含的单词间的编辑距离，计算出的字符串之间的相似度。而从直觉上来看，句子的文本相似度越高，其语义也就越相似，因此可以用来作为训练数据。本项目中使用 fuzzywuzzy 库进行相关特征计算，具体代码请参见 feature.py 中 dist_features()方法。

主题模型特征

项目中尝试使用 SVD, NMF, LDA 三个主题模型。为了比较不同主题模型，以及同一主题模型中主题数目对结果的影响，需要一种可行的比较方法。但由于主题向量难以直观的进行可视化，因此本项目中，我尝试使用不同配置的模型产生的向量进行距离特征提取，再利用提取到的距离特征训练分类模型，最终通过比较分类模型的效果，来比较主题模型的优劣。

比较过程中使用的模型是 lightGBM，参数使用的是默认值，与基准模型相同，比较结果如下：

topic model	logloss for train set	logloss for test set	time cost (s)
TruncatedSVD_50	0.5310311505349541	0.5665983960828199	3.461020
TruncatedSVD_100	0.5117958128700166	0.5571301109523962	6.968017
TruncatedSVD_200	0.5001674887803823	0.5495351528648251	15.738045
TruncatedSVD_300	0.4988523688347255	0.5380456856277878	20.068002
NMF_10	0.5539553510150845	0.5957539627244194	4.708004
NMF_20	0.5303164085755014	0.5759068008264504	11.239999
NMF_30	0.5205100335782367	0.5631362600287827	32.194003
LatentDirichletAllocation_10	0.5389515497446643	0.575780013394681	133.117587
LatentDirichletAllocation_20	0.5326364405869699	0.5718166521483227	131.473919

表 4

表 4 可以看到，主题模型中，SVD，主题数量为 300 时效果最好，而 NMF 和 LDA 想过相差不大。而且 NMF 效果略好于 LDA。而 SVD 中，主题数量越多，效果越好。

但同时，记录了主题模型的训练时间，见表 4 最后一列。在进行比较时，考虑到比较效率，并没有用全部数据进行试验，仅仅使用了 50000 条数据。考虑实际数据在 40 万条数据。且算法的复杂度都比较高。因此较为耗时的 LDA 在最后并没有使用。

主题模型比较的代码请参见 data_check5.ipynb，实现代码请参见 feature.py

句子间距离特征

根据 word2vec 模型，tf-idf 模型和上面提到的主题模型获取到的向量结果，可以使用各种相似度距离计算方法计算出句子对的相似度特征。在可视化部分也对一些特征进行了可视化分析。

句子间距离特征的计算在前面已经介绍过。在本项目中，使用各种方法把句子转化为向

量后，再使用 scipy 库中的 spatial.distance 库来进行各种距离的计算，包括余弦相似度，曼哈顿距离，雅卡尔指数，堪培拉距离，欧几里得距离，闵可夫斯基距离以及相异系数。

具体代码请参见 feature.py 中 add_dist_for 方法。

句子向量化

句子距离算法确定后，就需要将句子进行向量化，然后使用距离计算方法获取特征。同时，一些句子向量本身也可以作为特征。

本项目中使用的句子向量化方法包括如下几个方面

Word2vec 获取的向量

通用 Word2vec

前面介绍过，google 公布了一个 Google News vectors negative300 的 WMD 模型，可以用来进行 word2vec 计算。本项目中使用 gensim 库，加载该模型，进行计算。

基于预测数据的 word2vec

本项目中，加入了使用预测数据本身构建的 word2vec 模型计算的向量。模型的构建是在特征生成过程中，取得预测数据，构建一个语料库，再根据语料库训练 word2vec 模型，最后使用该模型进行向量化。具体代码请参见 feature_mp.py 中 local_vec_feature 方法下代码。

主题模型

本项目中尝试使用主题模型如表 4 所示。为了检验不同主题模型的效果，使用默认参数的 lightGBM 算法根据不同主题模型得到的向量直接训练模型，并检验模型效果。发现 TruncatedSVD 模型当主题数目设为 300 时效果最好，同时该模型训练速度快。

主题数量为 300 的模型结果效果好，可以用作距离计算。但若直接将其加入最终特征数据中，将引入 600 个新的列，会影响模型训练速度，因此另外训练了主题数为 25 的模型加入结果中。

NMF 模型效果也不错，但训练速度较慢，最终只训练了 NMF30，计算了距离特征，并也直接将其加入最终特征数据集里。

LDA 效果略逊于 NMF，且该算法速度很慢，很难应用于大数据集上，因此本项目最后没有使用该特征。

主题模型相关代码请参见 feature_mp 中 topic model 注释下的代码。

词性数据

本项目中，考虑词性统计数据。并使用 nltk 库中 wordnet 工具进行句子中单词的词性分

析。

由于细分的词性很多，而有些词性很少用于句子中，因此在使用词性数据前，我取少量数据统计了一下各种词性的词出现的频率，最终发现如下一些词性的词出现频率较大：WP, VB, DT, NN, IN, RB, TO, JJ, WRB, MD, PRP, CC, VBP

因此对于每个语句，统计这些词性的词的数量，将其转化成向量，计算各种距离特征，同时直接将这些数据加入特征数据集中，直接进行计算。

图特征

在以上特征获取后，可以得到一个粗略的句子间相似关系的结果，再将句子看作节点，统计不同节点间的关系。并像前面介绍的那样，将能够通过相似关系连接的所有节点都看作相似的句子，然后查看要判断的两个句子是否属于同一个联通的节点群中，从而判断句子是否相似。

执行过程

由于该项目属于分类问题，开始选取了 4 中进行分类的模型。在模型训练过程中要进行两方面工作：1，分别对每个模型进行参数调优。2，从四个模型中选择最优的模型。

模型调优：

参数调节过程使用了 sklearn 库中的 GridSearch 工具进行。对四个模型，分别使用 GridSearch 尝试模型中各种参数组合的结果。这部分的相关代码在 data_check.ipynb 文件中。

参数调节过程中使用参数如下：

GBDT 模型调优所用的参数已经在上一部分的方法与技术小节中介绍过，不再赘述。

Random Forest 中调节的参数包括：n_estimators，max_depth，min_samples_split,，max_features，min_samples_leaf。其中 max_features 参数代表训练过程中每棵决策树训练时使用的特征数量。默认是"auto",意味着划分时最多考虑 N 平方根个特征；如果是"log2"意味着划分时最多考虑 log2N 个特征；如果是"sqrt"或者"auto"意味着划分时最多考虑 N 平方根个特征。如果是整数，代表考虑的特征绝对数。如果是浮点数，代表考虑特征百分比。其中 N 为样本总特征数。

对 SGD 模型使用的参数包括：max_iter，tol，penalty，loss。其中

Max_iter 表示最大的训练迭代次数。

tol 表示迭代停止标准，如果迭代过程中损失函数值缩小的数值小于 tol 时，即可终止迭代。

penalty 表示惩罚方式,字符串型；默认为’l2’;其余有’none’,‘l1’,‘elasticnet’。

loss 表示损失函数选择项，字符串型；默认为’hinge’即 SVM；'log’为逻辑回归。

对逻辑回归模型的训练参数包括：

penalty 参数可选择的值为"l1"和"l2".分别对应 L1 的正则化和 L2 的正则化，默认是 L2 的正则化。

solver 参数决定了我们对逻辑回归损失函数的优化方法，有 4 种算法可以选择，分别是

a) liblinear：使用了开源的 liblinear 库实现，内部使用了坐标轴下降法来迭代优化损失函数。

b) lbfgs：拟牛顿法的一种，利用损失函数二阶导数矩阵即海森矩阵来迭代优化损失函数。

c) newton-cg：也是牛顿法家族的一种，利用损失函数二阶导数矩阵即海森矩阵来迭代优化损失函数。

d) sag：即随机平均梯度下降，是梯度下降法的变种，和普通梯度下降法的区别是每次迭代仅仅用一部分的样本来计算梯度，适合于样本数据多的时候。

对 LightGBM 调整的参数包括：

feature_fraction：默认值=1.0, 0.0 < feature_fraction < 1.0, 也称 sub_feature, colsample_bytree。如果 feature_fraction 小于 1.0, LightGBM 将会在每次迭代中随机选择部分特征. 例如, 如果设置为 0.8, 将会在每棵树训练之前选择 80%的特征，该参数可以用来加速训练，也可以用来处理过拟合。

bagging_fraction：0.0 < bagging_fraction < 1.0, 也称 sub_row, subsample。类似于 feature_fraction, 但是它将在不进行重采样的情况下随机选择部分数据。可以用来加速训练，也可以用来处理过拟合

bagging_freq：也称 subsample_freq。bagging 的频率为 0 意味着禁用 bagging. k 意味着每 k 次迭代执行 bagging

lambda_l1:默认为 0,也称 reg_alpha，表示的是 L1 正则化,double 类型

lambda_l2:默认为 0,也称 reg_lambda，表示的是 L2 正则化，double 类型

实际调参中，并没有找到较好的 lambda_l1 和 lambda_l2 值，最终仍使用默认值构建模型。

cat_smooth： 默认值为 10，用于分类特征。可以降低噪声在分类特征中的影响, 尤其是对数据很少的类别

具体的训练代码可以参考 data_check.ipynb 和 data_check2.ipynb 文件。基本的步骤是对各个模型使用 GridSearch 以及训练数据进行最优参数的挑选。最后使用测试数据集比较四个模型得到的最优结果的性能。

在最后模型性能比较的过程中我们使用了 logloss 来评价模型的性能。最终比较结果如图 12。每个条形图中的两个条分别表示模型对训练数据进行预测和对测试数据进行预测获得的 logloss。具体指标划分可以参见图例中的说明。纵坐标为各个指标的分数。总体的 logloss 分布在[0, 15]之间。

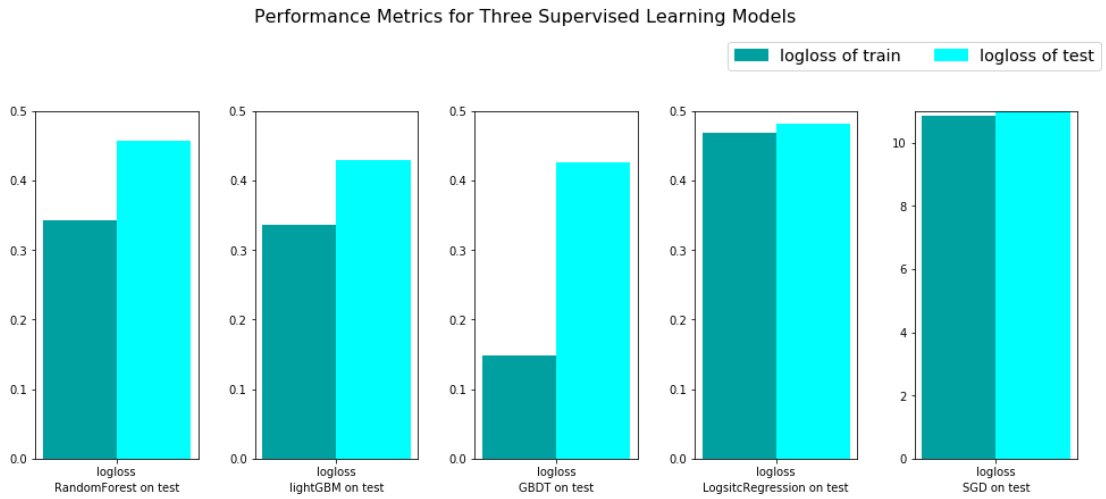

图 5-3-1

从最后结果的比价中可见，性能最好的还是 GBDT 经过调优后的结果。而最终该模型的调优参数如下：

learning_rate =0.1， n_estimators =140， max_depth =14，min_samples_split =550，max_features ='sqrt'， subsample=0.85

有趣的是，在基准模型选择时，五个算法中效果最好的是 lightBGM。而最终调优后发现，lightBGM 性能提高不多，而最终效果最好的仍是 GBDT 模型。

而其他模型的最终结果请参见 data_check.ipynb 和 data_check2.ipynb 文件。

特征的 iv 值请参加附录 1

结果

模型的评价与验证

最终模型是在项目开始时选取的几个模型算法进行参数调优后再选择最优的模型获得的。整个选择过程包括：模型调优和最终模型选择。初始模型选取了四个典型的用来进行分类的模型，包括：随机森林，GBDT，SGD，逻辑回归分类和 lightGBM 模型。

然后需要对每个模型进行参数调优，以获得每个模型中最优的模型结果。在这个过程中，我们使用 sklearn 中的 Grid Search 调试每个模型的参数，最后得到每个模型的最优结果。具体的调试过程和使用的参数可以参考方法部分的执行方法的小节，具体代码在 data_check.ipynb 文件中。

最后，对几个模型中使用 logloss 作为评价标准，并可视化各个最优模型所得到的对应指标结果。可视化结果的可视化如图 5-3-1 所示。具体的代码参考 data_check.ipynb 文件。

最终获得了用来分类的模型结果。最终模型使用的是 GBDT 算法，其参数调优结果如下：

learning_rate=0.1, n_estimators=150, max_depth=17, min_samples_split=250, max_features = 'sqrt' , min_samples_leaf=20, subsample=0.9

而模型的性能结果为：

Model	Train	Test
GBDT	0.14864674232915706	0.426406221610159

表 4

我认为模型结果比较合理，与项目开始时定义的基本模型（表 3）相比，最终模型的性能有明显提高。说明模型的调优和选择是合理的。

合理性分析

从上一章节中已经描述了模型调优和最终模型选择。初始模型选取了四个典型的用来进行分类的模型，包括：随机森林，GBDT，SGD 和逻辑回归分类模型。

然后需要对每个模型进行参数调优，这个过程中使用了 Grid Search 来筛选最优参数组合。在最后比较四个模型，得到了最终的模型。

而比较了基准模型和最终得到的模型的结果，可以看到最终模型比基准模型的效果要好，选定的三个对比指标都有所提高。

项目结论

结果可视化

总结本次项目，总共包含这几个步骤：数据特征分析与可视化，初始模型生成，模型训练以及模型选择。在这几个步骤中，运用了训练模型的相关技术，包括：数据准备与清理，变量生成，参数调优，最终模型选择这几个步骤。

数据的准备和清理过程主要是利用 pandas 库，对读入的多维数据进行异常值处理。主要是去除了数据中 none 值或无限大的数值。

变量生成的过程中主要采用了可视化方法，将变量的分布情况，以及目标字段的均值分布在各个变量上展示了出来，分析了使用的变量对于分类是否有明显的贡献。而生成的变量包括：句子字符串长度相关变量，句子向量化后向量间距离的相关变量，句子词移距离相关变量，以及句子模糊匹配产生的相关变量。具体变量和可视化结果分析请参见前文中可视化的章节。在这一部分中，使用了 matplotlib 进行了数据的可视化，代码参见 data_analyse.ipynb 文件中代码。同时，为了产生相关特征，使用了自然语言处理的相关工具，包括词移距离计算工具 gensim，模糊匹配工具包 fuzzywuzzy，自然语言处理工具包 nltk 等，具体代码请参考 features.py 文件。

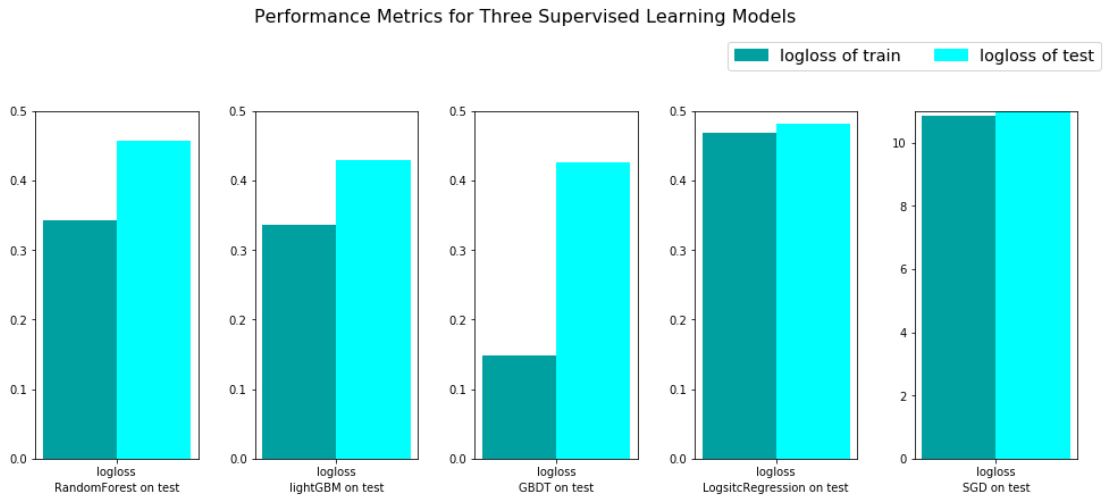

图 6-1-1

参数调优过程使用了 Grid Search 工具，针对各个模型参数，对模型进行了调整，得到了各个算法模型的较好结果。最后对结果进行了可视化，如图 13 所示。从结果中可以看到，对比基准模型数据（表 3），调试后各个模型的性能指标都有所提高。

最优模型的选取，通过图 13 的可视化分析，可以看到，比较四个模型，GBDT 的效果是最好的，该模型在测试数据上的结果是最好的，而且训练数据集的结果与测试数据集的结果差异不大，没有明显的过拟合现象。因此选用该模型作为最终模型。

对项目的思考

整个项目的流程在上一章节中已经介绍。

个人觉得项目中比较有趣的部分包括以下两点：

1. 各个变量可视化过程。在该过程中需要考虑如何可视化才能反映变量对目标字段是否有区分作用，因而使用了目标字段的平均值在变量上的分布情况来反映其区分效果。同时，还需要考虑变量本身的分布情况，以及如何转化才能有明显的可视化效果，因而需要对变量进行各种转化。例如句子长度比例的分布范围不对称，需要进行对数操作后才能获得较好的分布范围，也更有助于可视化。最后，可视化的过程其实也是一个变量提取的过程，例如句子长度比例的变量，在找到比例对数的表现形式后，该比例对数其实也是能够反映比例特征的最好的变量。
2. 寻找特征的过程。在这一过程中可以了解各种自然语言处理相关的技术，并能够不断尝试新的算法和技术。有助于了解更多的机器学习的相关算法和技术。
3. 可以考虑使用基于图的特征。语句间的相似可以看做是语句间的一种联系。而语句本身可以看做一个实体。这样可以构建一种图结构，并在图结构的基础上预测不同点间的紧密程度。

需要作出的改进

改进方向：

1. 尝试使用更多的自然语言处理相关的特征。
2. 对字符串进行多样化的分割，例如以空格分割；以空格加特殊字符分割；分割后去除停用词；分割后保留标点符号，标点符号视作独立单词等等

引用

Quora Question Pairs 背景信息：https://www.kaggle.com/c/quora-question-pairs

Matplotlib visualization：

Matplotlib: A 2D Graphics Environment：

http://aip.scitation.org/doi/pdf/10.1109/MCSE.2007.55?class=pdf

Beginning Python Visualization: Crafting Visual Transformation Scripts：

https://books.google.com/books?hl=zh-CN&lr=&id=N1InCgAAQBAJ&oi=fnd&pg=PP3&dq=python+visualization+matplotlib&ots=9GN8i96vNs&sig=wT6Oa4_ehjizHz2hnTpM0teGM-w#v=onepage&q=python%20visualization%20matplotlib&f=false

Python matplotlib 数据可视化：

https://absentm.github.io/2017/03/18/Python-matplotlib-%E6%95%B0%E6%8D%AE%E5%8F%AF%E8%A7%86%E5%8C%96/

评价指标：

Log Loss： http://wiki.fast.ai/index.php/Log_Loss

对数损失函数(Logarithmic Loss Function)的原理和 Python 实现：

https://www.cnblogs.com/klchang/p/9217551.html

Accuracy, Precision, Recall or F1：

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

机器学习算法：

Understanding Logistic Regression in Python：
https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

The Random Forest Algorithm：

https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

GBDT：梯度提升决策树

https://www.jianshu.com/p/005a4e6ac775

NLP 技术：

Word2vec： https://link.jianshu.com/?t=https://arxiv.org/pdf/1309.4168.pdf

自然语言预处理：https://wordnet.princeton.edu

主题模型：

https://blog.csdn.net/qq_39422642/article/details/78730662

LDA: https://blog.csdn.net/weixin_40604987/article/details/79615968

SVD: http://www.cnblogs.com/pinard/p/6251584.html

NMF: https://blog.csdn.net/u011089523/article/details/77340476

图特征：https://www.kaggle.com/divrikwicky/semi-magic-postprocess-0-005-lb-gain

附录

num	feature name	importance score
1	cosine_distance_TruncatedSVD_300	0.05818
2	fuzz_partial_ratio	0.050443
3	norm_wmd	0.037275
4	euclidean_distance	0.034694
5	fuzz_qratio	0.032991
6	common_ratio	0.024068
7	minkowski_distance_local_word2vec	0.016989
8	minkowski_distance	0.015312
9	fuzz_token_sort_ratio	0.01388
10	wmd	0.013279
11	cosine_distance_NMF_30	0.011844
12	fuzz_partial_token_sort_ratio	0.011606
13	cityblock_distance	0.010838
14	minkowski_distance_TruncatedSVD_300	0.010087
15	cityblock_distance_local_word2vec	0.009967
16	braycurtis_distance	0.009471
17	braycurtis_distance_TruncatedSVD_300	0.008376
18	q1_TruncatedSVD_25_8	0.007815
19	q1_TruncatedSVD_25_12	0.007407
20	braycurtis_distance_local_word2vec	0.007339
21	fuzz_token_set_ratio	0.007298
22	q1_TruncatedSVD_25_17	0.007169
23	skew_q2vec_local_word2vec	0.006681
24	q1_TruncatedSVD_25_14	0.006574
25	cosine_distance	0.006506
26	q1_TruncatedSVD_25_20	0.006433
27	q2_TruncatedSVD_25_14	0.006379
28	len_q1	0.006254
29	q1_TruncatedSVD_25_21	0.006246
30	q1_TruncatedSVD_25_2	0.006243
31	q1_NMF_30_23	0.00606
32	fuzz_WRatio	0.006055
33	q2_TruncatedSVD_25_20	0.006032
34	canberra_distance_TruncatedSVD_300	0.006032
35	q2_TruncatedSVD_25_8	0.005998
36	euclidean_distance_TruncatedSVD_300	0.005957
37	q2_TruncatedSVD_25_13	0.00583
38	q2_TruncatedSVD_25_18	0.005796

39	q2_TruncatedSVD_25_17	0.005778
40	canberra_distance_local_word2vec	0.00576
41	canberra_distance	0.005467
42	q1_TruncatedSVD_25_7	0.005447
43	q1_TruncatedSVD_25_19	0.005436
44	cityblock_distance_TruncatedSVD_300	0.005393
45	q1_TruncatedSVD_25_1	0.005339
46	q1_TruncatedSVD_25_3	0.005238
47	kur_q2vec_TruncatedSVD_300	0.00519
48	euclidean_distance_local_word2vec	0.005184
49	skew_q1vec_local_word2vec	0.005157
50	q2_TruncatedSVD_25_10	0.005156
51	canberra_distance_pos_vec	0.005114
52	q2_TruncatedSVD_25_23	0.005084
53	q1_TruncatedSVD_25_24	0.005006
54	q2_TruncatedSVD_25_12	0.005003
55	q1_TruncatedSVD_25_13	0.004962
56	q1_TruncatedSVD_25_6	0.004959
57	q2_TruncatedSVD_25_2	0.004798
58	q2_TruncatedSVD_25_1	0.004789
59	q2_TruncatedSVD_25_21	0.004789
60	cityblock_distance_NMF_30	0.004685
61	q2_TruncatedSVD_25_6	0.004647
62	cosine_distance_local_word2vec	0.004622
63	q1_TruncatedSVD_25_22	0.004606
64	q2_TruncatedSVD_25_4	0.004588
65	skew_q2vec	0.004569
66	q1_NMF_30_22	0.00453
67	q2_NMF_30_23	0.004522
68	len_q2	0.004519
69	q1_TruncatedSVD_25_18	0.004508
70	q2_NMF_30_22	0.004387
71	q1_TruncatedSVD_25_16	0.004338
72	kur_q1vec_NMF_30	0.004311
73	len_q_ratio_ln	0.00431
74	kur_q2vec_local_word2vec	0.004287
75	euclidean_distance_NMF_30	0.004283
76	q2_TruncatedSVD_25_3	0.00422
77	q2_TruncatedSVD_25_24	0.004215
78	q1_TruncatedSVD_25_4	0.004178
79	q2_TruncatedSVD_25_16	0.00413
80	q1_TruncatedSVD_25_11	0.004123
81	q2_TruncatedSVD_25_7	0.004122

82	q2_TruncatedSVD_25_19	0.004116
83	kur_q2vec	0.004104
84	q1_TruncatedSVD_25_0	0.004054
85	braycurtis_distance_NMF_30	0.00405
86	skew_q2vec_TruncatedSVD_300	0.004049
87	q2_TruncatedSVD_25_0	0.004039
88	kur_q1vec_TruncatedSVD_300	0.004031
89	skew_q1vec_TruncatedSVD_300	0.003959
90	q1_TruncatedSVD_25_15	0.003846
91	q1_TruncatedSVD_25_5	0.003805
92	kur_q1vec_local_word2vec	0.003799
93	minkowski_distance_NMF_30	0.003792
94	q2_TruncatedSVD_25_15	0.003773
95	q1_TruncatedSVD_25_23	0.003714
96	q2_TruncatedSVD_25_9	0.0037
97	skew_q1vec_NMF_30	0.003694
98	canberra_distance_NMF_30	0.003686
99	q2_TruncatedSVD_25_11	0.003648
100	cosine_distance_pos_vec	0.003647
101	skew_q2vec_NMF_30	0.003616
102	kur_q1vec_pos_vec	0.003584
103	q1_TruncatedSVD_25_10	0.003572
104	len_char_q_ratio_ln	0.00356
105	len_word_q2	0.003549
106	q2_TruncatedSVD_25_5	0.003533
107	q2_NMF_30_4	0.003522
108	q2_TruncatedSVD_25_22	0.003493
109	kur_q2vec_NMF_30	0.003486
110	kur_q1vec	0.003469
111	skew_q1vec	0.003457
112	q1_NMF_30_4	0.003453
113	q1_NMF_30_3	0.003397
114	len_word_q_ratio_ln	0.003362
115	q1_NMF_30_12	0.003356
116	q1_NMF_30_0	0.00335
117	q1_TruncatedSVD_25_9	0.003313
118	q1_NMF_30_29	0.003298
119	q2_NMF_30_29	0.003279
120	q1_NMF_30_2	0.00326
121	q1_NMF_30_13	0.003201
122	euclidean_distance_pos_vec	0.00318
123	len_char_q2	0.003068
124	q2_NMF_30_12	0.00305

125	q2_NMF_30_2	0.002892
126	q2_NMF_30_0	0.002887
127	q2_NMF_30_24	0.002871
128	len_word_q1	0.002833
129	skew_q1vec_pos_vec	0.002705
130	q1_NMF_30_11	0.002699
131	q2_DT	0.002662
132	q2_NMF_30_1	0.002586
133	skew_q2vec_pos_vec	0.002569
134	kur_q2vec_pos_vec	0.002563
135	q1_NMF_30_19	0.002489
136	q1_NMF_30_24	0.00248
137	q1_NMF_30_5	0.002479
138	q2_NMF_30_13	0.002462
139	q2_NMF_30_8	0.00246
140	q2_NMF_30_3	0.002432
141	q2_NMF_30_27	0.002427
142	braycurtis_distance_pos_vec	0.002396
143	q1_NMF_30_1	0.002372
144	q2_NMF_30_28	0.002291
145	q2_NMF_30_5	0.002256
146	minkowski_distance_pos_vec	0.002222
147	len_char_q1	0.002196
148	q2_NMF_30_11	0.002188
149	cityblock_distance_pos_vec	0.002152
150	diff_len	0.002125
151	q2_NMF_30_10	0.002124
152	q1_NMF_30_28	0.002103
153	q2_NMF_30_6	0.002087
154	q1_NMF_30_25	0.002038
155	jaccard_distance_pos_vec	0.002031
156	q1_NMF_30_15	0.002025
157	q2_NMF_30_18	0.002005
158	q2_NMF_30_16	0.001995
159	q2_NMF_30_25	0.001968
160	q1_NMF_30_18	0.001887
161	q2_NMF_30_21	0.001871
162	q1_NMF_30_21	0.00186
163	q1_NMF_30_17	0.001859
164	q2_NMF_30_19	0.001856
165	q1_NMF_30_10	0.001845
166	q1_NMF_30_26	0.00184
167	q1_NMF_30_27	0.00174

168	q1_NMF_30_8	0.001738
169	q1_NMF_30_6	0.00173
170	q2_NMF_30_17	0.001702
171	q1_NMF_30_9	0.001637
172	q2_NMF_30_26	0.001631
173	q1_NMF_30_16	0.001609
174	q2_NMF_30_9	0.001606
175	common_words	0.001594
176	q1_NMF_30_14	0.001582
177	q1_NN	0.001528
178	q1_NMF_30_20	0.001516
179	q2_NMF_30_14	0.00142
180	q2_NMF_30_7	0.001209
181	q2_NMF_30_15	0.001178
182	q2_NMF_30_20	0.001177
183	q1_DT	0.001141
184	q1_VB	0.001091
185	q2_NN	0.001038
186	q1_NMF_30_7	0.001025
187	q2_RB	0.000746
188	q1_IN	0.000657
189	q2_VB	0.00063
190	q1_RB	0.000592
191	jaccard_distance	0.000572
192	q2_WP	0.000571
193	q2_IN	0.000501
194	q1_WP	0.000411
195	jaccard_distance_local_word2vec	0.000389
196	q1_JJ	0.000388
197	q1_TO	0.000375
198	q2_JJ	0.000366
199	q2_MD	0.00033
200	jaccard_distance_NMF_30	0.000316
201	jaccard_distance_TruncatedSVD_300	0.000293
202	q1_MD	0.00027
203	q2_WRB	0.000268
204	q1_VBP	0.00026
205	q1_WRB	0.000258
206	q1_CC	0.000203
207	q2_VBP	0.000149
208	q2_PRP	0.000146
209	q1_PRP	0.000117
210	q2_CC	0.000084

211	q2_T0	0.000056
212	fuzz_partial_token_set_ratio	0