机器学习毕业项目

张高超

1. 问题的定义

1.1 项目概述

(Nature language process) NLP 是当前非常热门的主题，而比较句子相似度的任务是这一主题下很重要的话题，会被应用如问答，信息提取，文档相似度比较等各个方面。在 NLP 中有很多基础的模型，如词袋模型 [1]来表示一段文档的向量，TFIDF [2]来表示词频和词的重要程度，主题模型，如 PCA [3]主成分分析来提取主题，LDA[4]主题模型等，这些技术提供了 NLP 发展的重要基础。其次在词向量上，有 word2vec [5]，glove [6]，中文腾讯开源的语料库 [7]都给 NLP 分析数字化特征提取带来的重要的参考。

提取了有效的特征之后，在机器学习方面，也有很多相关的有效方法，如逻辑回归 Logistic Regression，GBDT，RF，xgboost 等等优秀的机器学习算法给 NLP 的处理提供了很多途径。

最近，随着深度学习的流行，深度学习网络模型在比较句子相似度上有了很大的发展，如 DSSM [8], DeepMatch [9], CDSMM [10], ARC-I [11], CNTN [12], LSTM-RNN [13] and Bi-CNN-MI [14]。

在 Quora 这个项目中，提供了比较丰富的样本数据，是一个探索文本相似度方向的优秀数据，而且互联网内容大部分还是泛文本内容，对于以后的学习和扩展有很大帮助。

1.2 问题陈述

这个项目可以被看做是一个二分类问题，即两个问题是否相似，我们通过 Qura dataset [15]下载数据，可以通过上面的机器学习或深度学习模型，训练出一个模型，使得 f(q1,q2)映射到（0，1）的范围，这个数字越大，相似度越高，反之越低。其中包含比较复杂的情况，如句子的词很多相同但是逻辑上很大不同，或者词完全不同，但是是表示同一个意思，这些都需要我们测试不同的模型，找到最佳的模型来，在此项目使用 log loss 来作为损失函数，来测试模型的优劣。

1.3 评价指标

在这个项目中我使用 Kaggle 比赛中使用的评价函数，即 log loss。每一对问题对都会有一个 0 到 1 的相似度概率，具体的公司为：

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

N 是问题对总数，M 是分类标记数，yij 为 1 如果第 i 个数再第 j 个类里，否则为 0，pij 是预测第 i 个数再第 j 个类里的概率，具体的推导可以详见 wiki [16]。

2. 分析 2-4

2.1 数据的探索

训练集和测试集是有 kaggle 的官网提供，训练集有 21MB，测试集有 112MB，在训练集里面有 6 个属性：分别为 id（该问题对

的独立 ID），qid1，qid2（每个问题对应的独立 ID），question1，question2（问题 1 和问题 2 对应的文本内容）， is_duplicate（判断两个是否相似，相似为 1，否则为 0）。

下面是训练集的前 5 行：

	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

简单的对训练集进行数据分析可以得知：

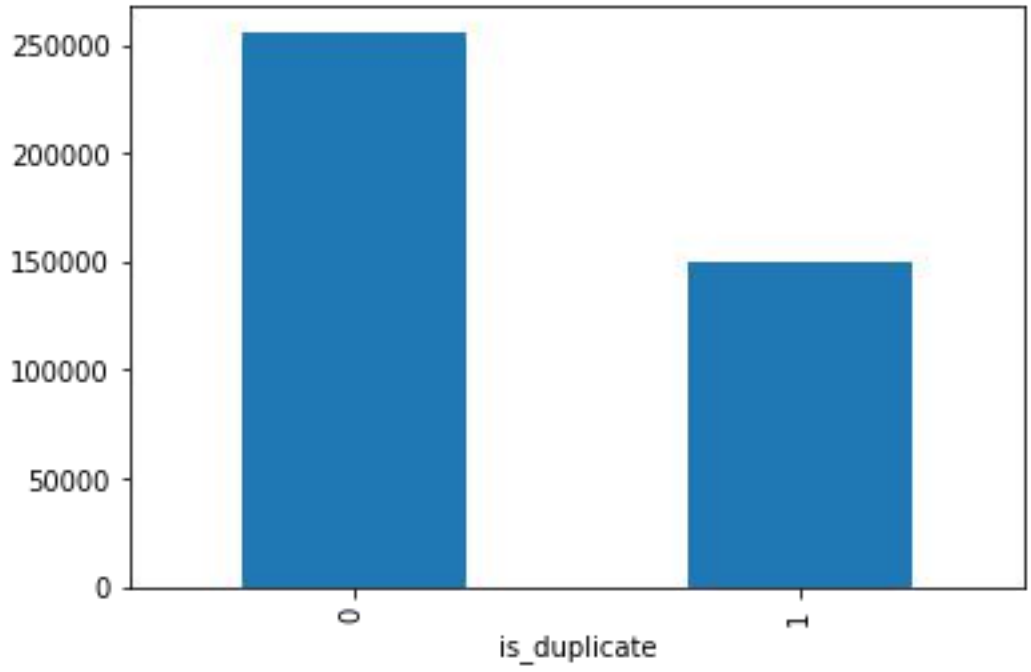
训练集有 404290 个数据；

重复问题对占比 36.92% ；

总问题数有 537933 条；

有 111780 问题出现多次。

下图是重复问题的分布：



下面是测试集的前 5 行：

	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?

测试集有 2345796 对。

接着我们需要探索问题和可能需要的特征，一些特征会在下面的可视化中有所展现：

1.首先是基本特征，即问题 1 和问题 2 的长度，字符数，两个问题的长度之差，字符数之差，单词数之差等等。

2.问题间的编辑距离，又称莱文斯坦距离(Levenshtein distance)30，包含了词间 的简单匹配距离比、部分匹配距离比、词体无序比等，通过使用 python Levenshtein 包去提取特征。

3.问题的疑问词统计，涉及的疑问词有 what、why、when、how、where、which 和 whose

4.词强度，计算两个问题间同时出现词所占的比值，与单独出现的词的词力(word power)，本项目通过开发 get_same_word_weight 函数提取相同词比重，其可视化结果参考下一节。

5.问题的情感与主观量化

6.问题的语义词统计，涉及的词类标记(POS tag)有 'IN'(介词)、'JJR'(比较 级形容词)、'JJS'(最高级形容词)、'MD'(情态助词)、 'PDT'(前限定 词)、'PRP'(代词)、'PRP$'(代词所有

格)、'RB'(副词)、'RBR'(比较级副 词)、'RBS'(最高级副词)、'RP'(量词)、'VB'(动词一般现在时)、'VBD'(动词一般过去时)、'VBG'(动词一般进行时)、'VBN'(动词过去分词)、 'VBP'(动词第一人称单数)、'VBZ'(动词第三人称单数)。

7.问题的停用词统计，包括两个问题各自停用词的频数与比率，主要有
'the','a','an','and','but','if','or','because','as','what','which','this','that','these','those','then',
'just','so','than','such','both','through','about','for','is','of','while','during','to','What','Which',
'Is','If','While','This'等停用词。

8.问题的实体分析，包括了杰卡德系数(Jaccard Index)31、两个问题的实体数目与实体标签数。

9.一个问题视为一个节点，一条线所连接的两个点表示数据集中的两个问题为一个问题对，以此构建图，在此图的基础上得到每个问题对点与线的量化值。

10.问题的相似性分析，计算通过 gensim32 和 tfidf 得到的词向量间的相似性。

11.问题的逆文档频率(Inverse Document Frenquency)统计值，与 使用 GoogleNews、GloVe 和基于问题集预训练得到的 Word2Vec 向量所得到的相似值，本项目通过 sklearn 的

tfidfvectorizer 包生成每个问题的向量然后计算相似度提取特征。

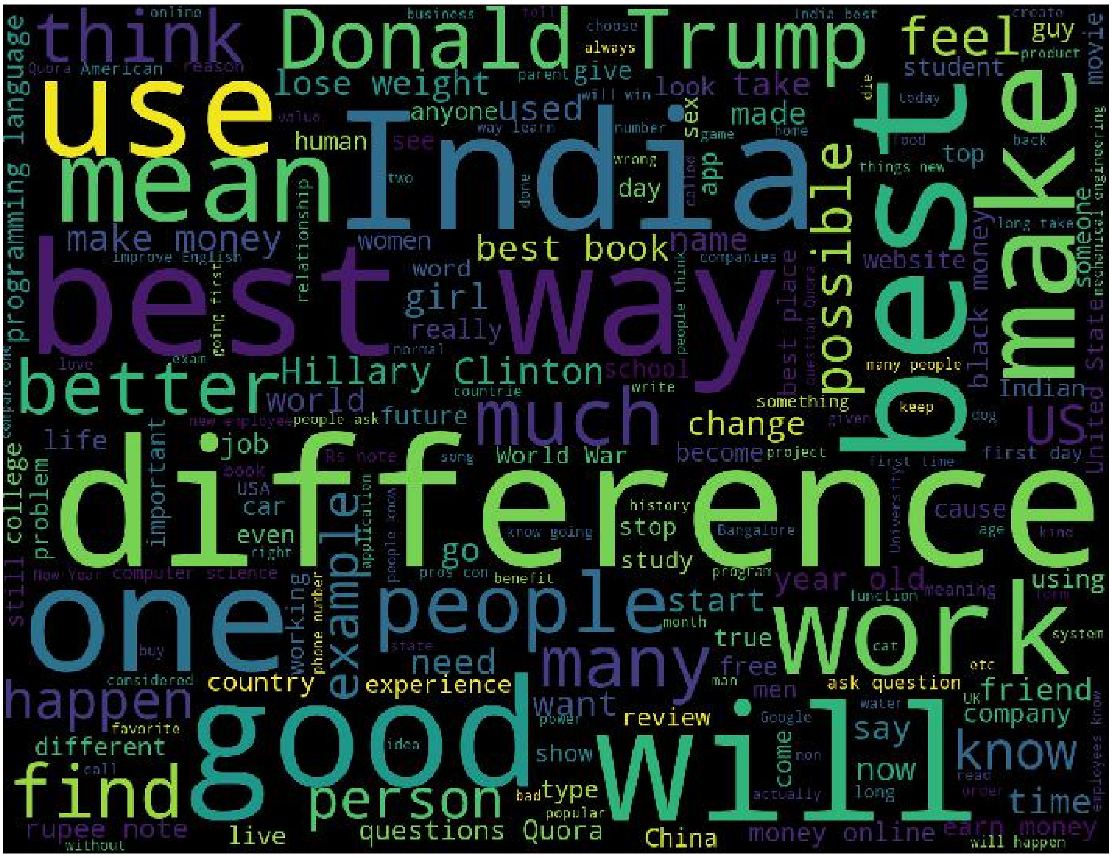
12.使用基于问题集预训练得到的 Word2Vec 所求得多种距离值，有余弦距离(cosine Distance),曼哈顿距离(Manhattan Distance),欧几里得距离(Euclidean Distance)，本项目使用 GoogleNews-vectors-negative300.bin.gz 词向量，得到 doc2vec 特征，然后计算相关距离特征。

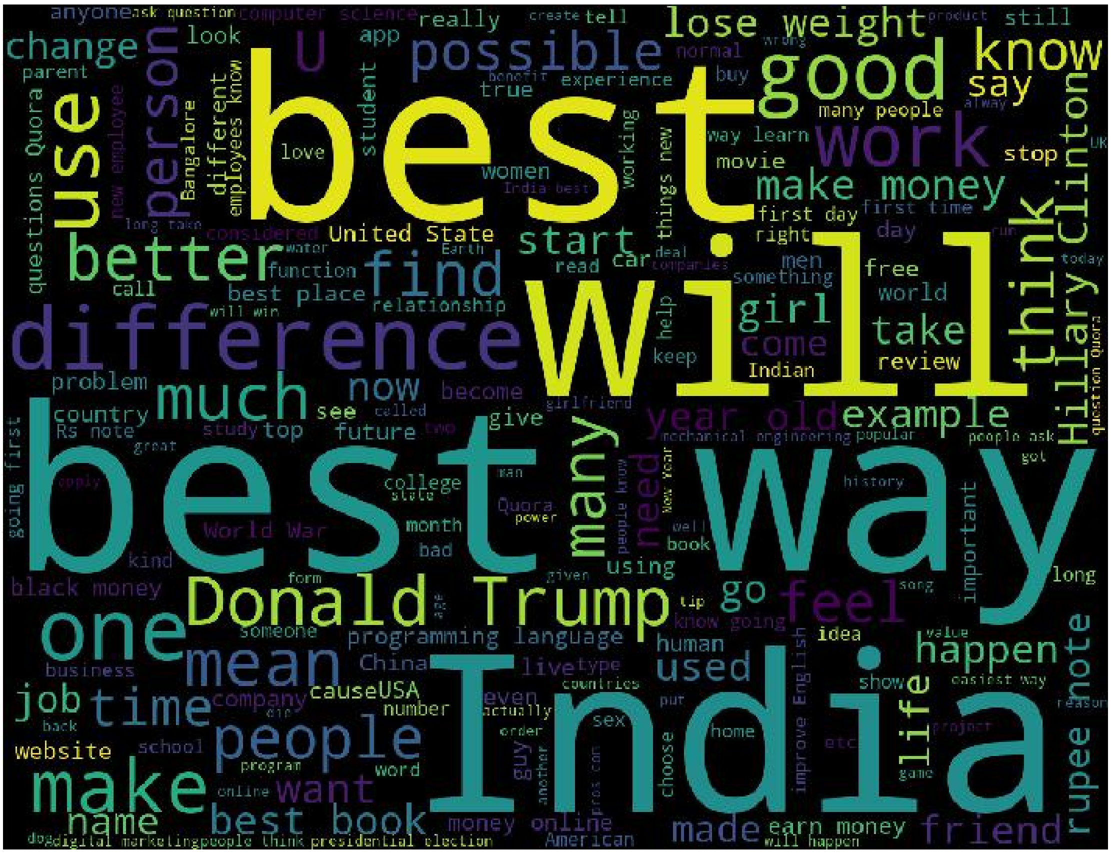
13.主题模型可以将文档集 中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题（分布）出来后，便可以根据主题（分布）进行主题聚类或文本分类。同时，它是一种典型的词袋模型，即一篇文档是由一组词构成，词与词之间没有先后顺序的关系，使用预训练的 tf-idf vectorizer 分别转化两个问题，然后使用 TruncatedSVD/NMF/LDA 得到维度各为 20 的特征集，并计算余弦相似度，L1 距离和 L2 距离，具体结果见下 3.2。

2.2 探索性可视化

通过 wordcloudgraph 包，我们可以对数据中问题描述的概况有一个总体的认知，一个词出现的概率越大，在图中的占比越大，字体越大，我们通过把问题分解为两个词库并调用 word cloud 库来产生图。

如下图：

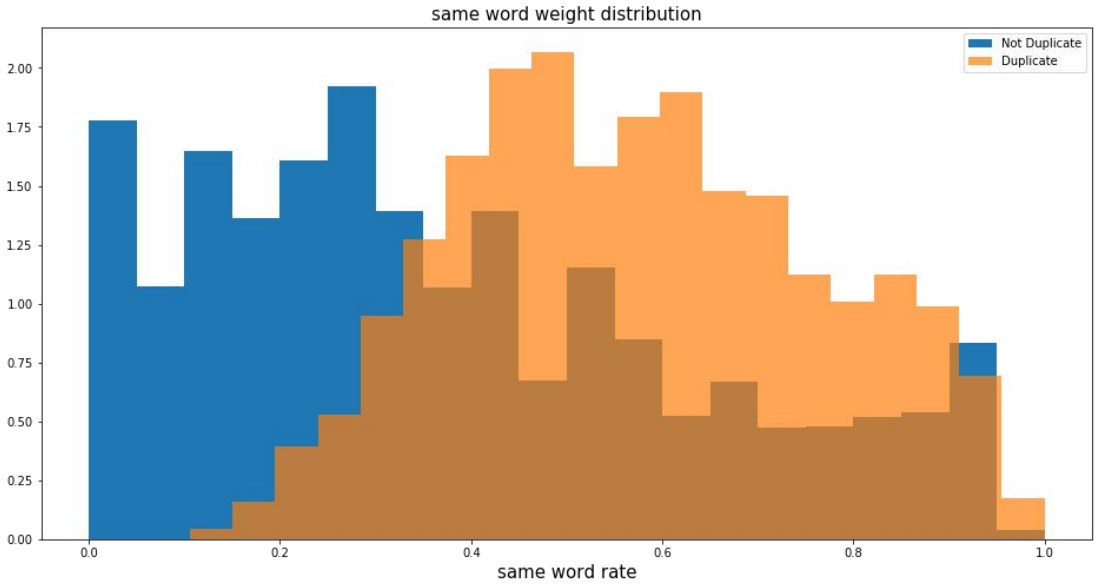

difference
best
India
way
will
good
one
work
make
best
use
mean
people
many
find
person
better
much
happen
Donald Trump
think
feel
Hillary Clinton
lose weight
make money
best book
possible
know
time
example
questions Quora
China
money online
rupee note


best
will
best way
India
Donald Trump
difference
good
work
know
use
better
find
much
one
mean
people
make
possible
person
think
Hillary Clinton
lose weight
make money
many
need
year old
example
happen
rupee note
best book
friend
made
time
job

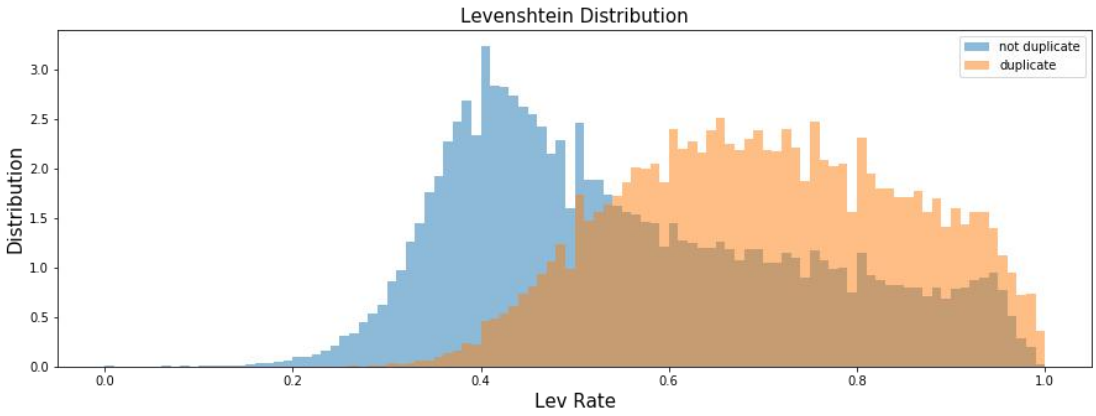
可以看出问题1和问题2的主要词分布，主要有best way，best，will，difference，India等等词汇。

字符分布，可以看出尖峰点在45个字符附近，平均值在55个平均值附近。在150个字符数处有一个小的尖峰，可能是quora网站对于问题的大小在150个字符处有一个限制，所以会有一个尖峰。

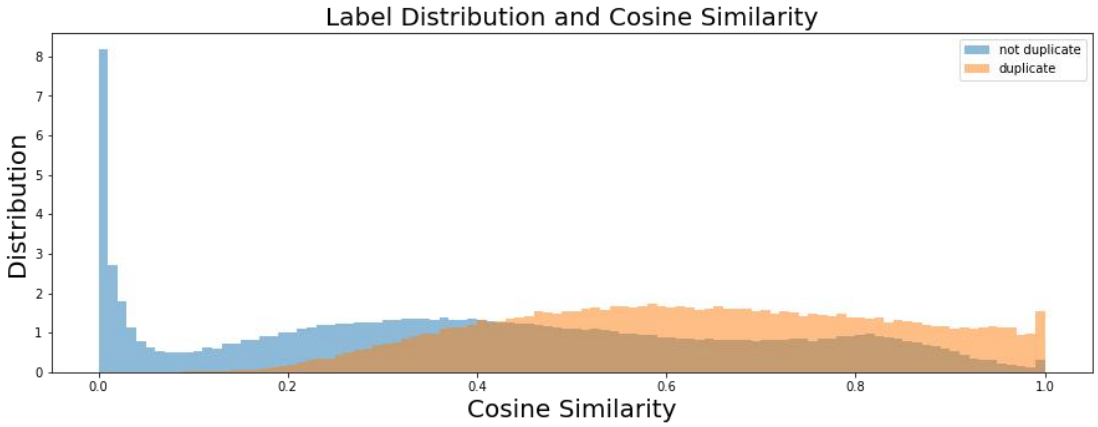
下图是word share ratio 图，这个是从训练集来抽出特征，即共同词的占比，可以看出重复问题对的相同词比例总体上来说大于非重复问题对。



第四个图是tf-idf 图，更上面的word share 差不多，可以看出重复问题对的相同词比例总体上来说大于非重复问题对：



最后一个是计算余弦相似度，通过 tfidf 转换特征向量：



2.3 算法和技术

因为总的来说这个一个二分类问题，我将使用监督学习方法，首先我会做一些预处理 ，如删除句子前后的空格，把句子都变为小写，然后我会抽取相应的特征，如字符数，词数，word share，tf-idf share；我将测试以下算法：随机森林，逻辑回归，决策树算法，朴素贝叶斯，支持向量机，xgbt 等等算法，然后用验证集测试所有的算法，测试各个算法的效果，接着做出相应的调整和使用集成学习的方法找到最优解；

逻辑回归算法是一个回归模型，返回的是结果的概率，为 0-1 的范围；

决策树算法是通过树结构来在每个节点做二分类，然后在叶子节点返回；

随机森林算法是集合了集成学习的一种算法，他通过使用bagging 的方法，即通过选择训练数据的随机样本来进行训练，来产生不同的决策树，最后通过平均这个决策树的结果进行预测；

knn 是使用 k 个领域点，通过他们距离的平均去做预测；

简单贝叶斯算法是使用贝叶斯定理进行预测；

支持向量机通过最佳超平面找到对应的间隔面来确定二分类；

2.4 基准模型

使用随机森林模型，因为随机森林比较容易实现，而且有一个比较好的结果。

3 方法 3-5

3.1 数据预处理

首先去掉前面和后面的空格，然后把所有的字符变为小写，然后去掉所有的空值，并用 0 或 empty 来代替；然后使用特征工程提取相应的特征，如问题 1 的词数量，问题 1 的字符数，问题 2 的词数量，问题 2 的字符数，共有词比例。

3.2 执行过程

3.2.1 特征工程

1. 基本特征先使用特征工程提取相应的特征，如问题 1 的词数量，问题 1 的字符数，问题 2 的词数量，问题 2 的字符数，共

有词比例，范例如下：

	len_q1	len_q2	diff_len	len_char_q1	len_char_q2	len_word_q1	len_word_q2	common_words
0	66	57	9	66	57	14	12	10
1	51	88	37	51	88	8	13	4
2	73	59	14	73	59	14	10	4
3	50	65	15	50	65	11	9	0
4	76	39	37	76	39	13	7	2

2. 主题模型，使用预训练的 tf-idf vectorizer 分别转化两个问题，然后使用 TruncatedSVD/NMF/LDA 得到维度各为 20 的特征集，并计算余弦相似度，L1 距离和 L2 距离，范例如下：

cosine_distance	l2_distance	l1_distance	0	1	2	3	4	5	6	7
0.063406	1.879174e-01	6.330075e-01	2.240579e-02	2.669546e-02	-3.875142e-02	-3.067216e-02	5.870540e-02	-6.323612e-02	1.671744e-02	3.464102e-02
0.512164	9.818788e-02	3.297411e-01	-7.891656e-03	-1.619105e-02	2.107553e-02	1.016566e-02	-1.510418e-02	2.187982e-02	-2.909166e-02	3.323529e-03
0.213442	3.928053e-02	1.265264e-01	6.920414e-03	8.667128e-03	-9.124414e-04	-2.422969e-03	-3.442583e-03	9.018676e-04	-5.094534e-03	-8.380206e-03
0.563036	4.770827e-02	1.541957e-01	4.970049e-04	9.776609e-04	-3.900871e-03	4.751267e-03	4.506634e-03	-5.904406e-03	-7.696592e-03	-4.880743e-03
0.358203	1.261113e-02	4.002655e-02	-1.834605e-03	-2.772930e-04	1.407261e-03	1.398018e-03	-3.954879e-03	4.417108e-03	8.915010e-04	-1.150865e-03
0.189574	1.074144e-02	3.323053e-02	-1.201657e-03	-5.629415e-03	4.808181e-03	2.611626e-03	4.030632e-03	-1.194972e-03	2.082266e-03	-3.596382e-04
0.681348	7.494778e-02	2.600185e-01	1.648404e-02	8.518902e-04	2.160568e-02	-3.795588e-03	-3.729156e-03	3.498986e-04	6.651969e-03	6.793606e-03

3. 之后尝试[8]git 库中提供的特征进行改造修改尝试，使用 fuzzy 包提取相关特征，范例如下：

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
93	95	98	100	89	100	93
66	86	73	100	75	86	63
54	63	53	100	71	66	66
35	35	30	37	38	36	36
46	86	54	100	63	67	47

4. 使用 GoogleNews-vectors-negative300.bin.gz 词向量包，计算距离等相关特征，范例如下：

wmd	norm_wmd	cosine_distance	cityblock_distance	jaccard_distance	canberra_distance	euclidean_distance	minkowski_distance	braycurtis_distance
0.563226	0.244216	0.063406	4.869509	1.0	89.759695	0.356106	0.161325	0.179626
3.772346	1.368796	0.512164	14.195119	1.0	177.588090	1.012091	0.455910	0.592655
1.645910	0.660150	0.213442	9.201546	1.0	134.665397	0.653363	0.291829	0.343318
3.106745	1.263368	0.563036	14.894639	1.0	186.419017	1.061166	0.472624	0.639723
3.088408	1.062413	0.358203	11.895132	1.0	160.012510	0.846408	0.376750	0.472422

5. 同样通过python提供的Levenshtein包计算两个问题间的编辑距离，通过上面的特征图，这个特征也是有效，范例如下：

same_word_weight
0.166667
0.380952
0.444444
0.000000
0.600000

6. 通过python Levenshtein包计算编辑距离，提取特征，结果范例如下：

lev_ratio
0.448000
0.477064
0.606742
0.545455
0.677419

7. 最后通过TfidfVectorizer工具，计算词频向量，接着提取余弦相似度的特征，结果范例如下：

cos_sim
0.000000
0.614661
0.595018
0.000000
0.736774

接着是使用不同的算法进行训练测试，尝试了以下模型，然后进行训练，对应的参数和细节如下：

1.首先使用随机森林作为基本模型，使用如下超参数：parameters_rf = {'n_estimators':[16, 32, 64, 128, 512], 'random_state':[0]}进行gridsearch交叉验证；

2.使用决策树算法，使用 dt = DecisionTreeClassifier(max_depth = 100, random_state=10) 作为超参数；

3.使用 SVM 支持向量机算法,使用 svm = SVC(random_state=0, max_iter=500, probability=True) grid = {'C': [1e0, 1e3],'kernel':['linear', 'poly', 'rbf', 'sigmoid']}作为超参数；

4.使用逻辑回归算法，使用超参数 parameters_lr = {'solver':['sag'], 'C':[.01, 0.001, 0.0001], 'random_state':[0]}进行 gridsearch 交叉验证，得到最佳模型参数为('Best estimator: ', LogisticRegression(C=0.01, class_weight=None, dual=False,fit_intercept=True,intercept_scaling=1,max_iter=100, multi_class='ovr', n_jobs=1,penalty='l2', random_state=0, solver='sag', tol=0.0001,verbose=0, warm_start=False))。

5.使用 XGBOOST 算法，使用超参数 params = {}

params['objective'] = 'binary:logistic'

params['eval_metric'] = 'logloss'

params['eta'] = 0.02

params['max_depth'] = 8。

3.3 完善

通过尝试不同的超参数，和不同的集成学习策略进行不同训练，尝试不同的模型，发现有时候会过拟合。

4 结果

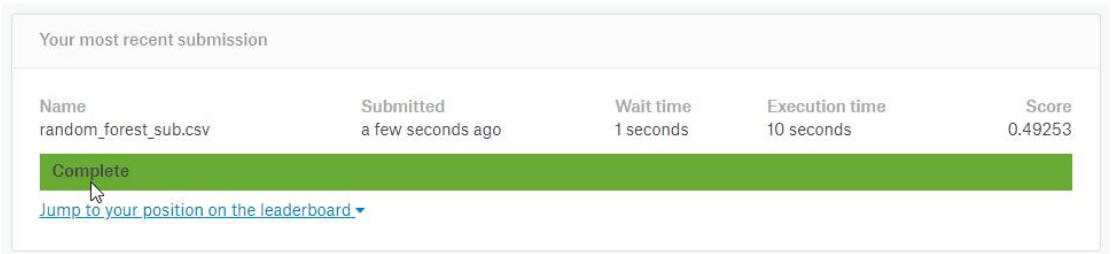
4.1 模型的评价与验证

随机森林算法：

验证集：

```
In  [114]: rand_forest_y_pred = cv_rf.predict_proba(x_valid)
           log_loss(y_valid, rand_forest_y_pred[:,1])
Out[114]: 0.4210279800717873
```

测试集：

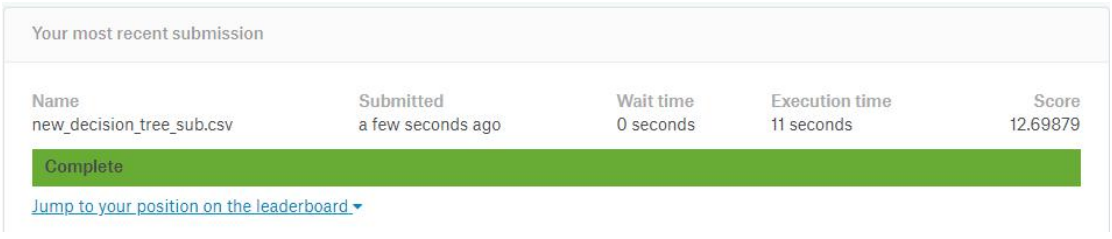


决策树算法：

验证集：

```
In  [121]: log_loss(y_valid, dt_y_pred)
Out[121]: 9.879748271566388
```

测试集：



支持向量机算法：

验证集：

```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed: 64.1min finished
{'kernel': 'sigmoid', 'C': 1.0}
SVM model accuracy: 0.680
```

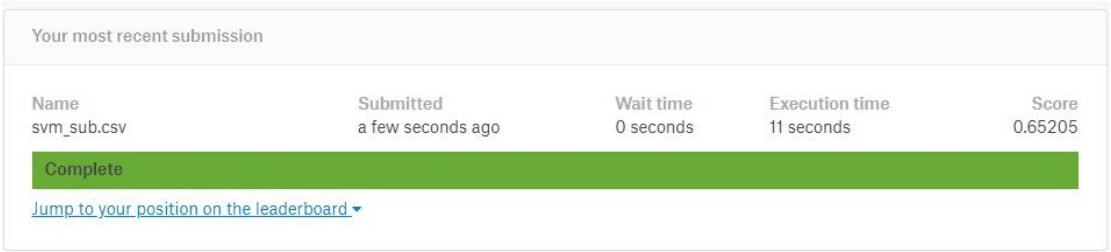
In [130]:
```
svm = SVC(random_state=0, max_iter=1000, probability=True, kernel='linear', C=1.0)
svm.fit(x_train, y_train)
```

Out[130]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=1000, probability=True, random_state=0,
  shrinking=True, tol=0.001, verbose=False)
```

In [131]:
```
svm_y_pred = svm.predict_proba(x_valid)
log_loss(y_valid, svm_y_pred[:,1])
```

Out[131]:
```
0.5940690946566914
```

测试集：



逻辑回归算法：

验证集：

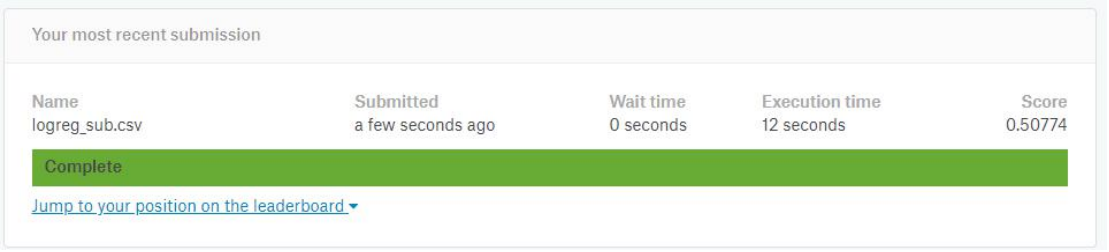
```
('Best estimator: ', LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=0, solver='sag',
          tol=0.0001, verbose=0, warm_start=False))
```

In [135]:
```
logreg_y_pred = cv_lr.predict_proba(x_valid)

log_loss(y_valid, logreg_y_pred[:,1])
```

Out[135]:
```
0.5444400123113126
```

测试集：

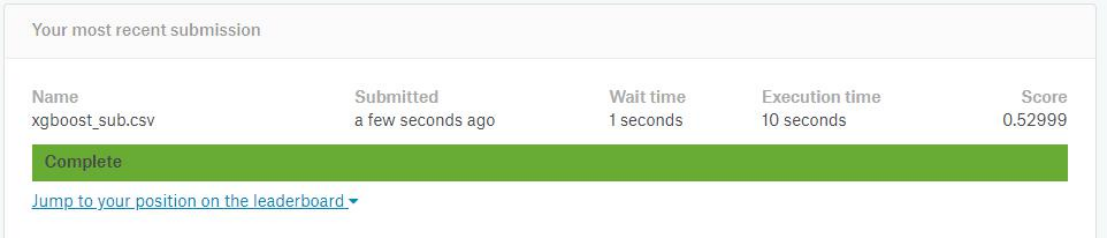


XGBoost 算法：

验证集：

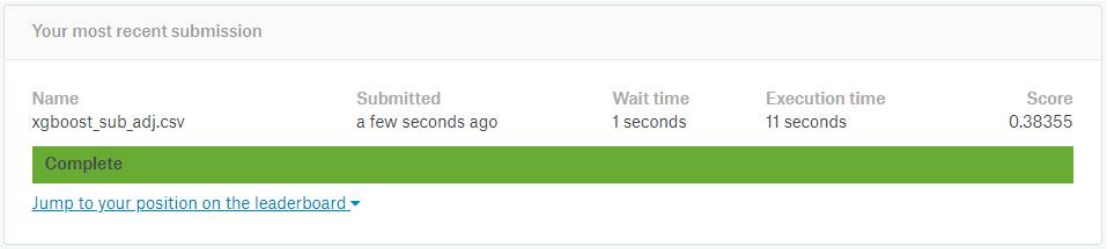
```
[10]    train-logloss:0.630805  valid-logloss:0.63183
[20]    train-logloss:0.590615  valid-logloss:0.592387
[30]    train-logloss:0.560666  valid-logloss:0.563131
[40]    train-logloss:0.537751  valid-logloss:0.540926
[50]    train-logloss:0.519679  valid-logloss:0.523527
[60]    train-logloss:0.505358  valid-logloss:0.509865
[70]    train-logloss:0.493797  valid-logloss:0.498901
[80]    train-logloss:0.484292  valid-logloss:0.489966
[90]    train-logloss:0.476801  valid-logloss:0.483023
[100]   train-logloss:0.470616  valid-logloss:0.477319
[110]   train-logloss:0.465233  valid-logloss:0.472448
[120]   train-logloss:0.460743  valid-logloss:0.468389
[130]   train-logloss:0.457045  valid-logloss:0.465147
[140]   train-logloss:0.453963  valid-logloss:0.462453
[150]   train-logloss:0.451229  valid-logloss:0.460048
[160]   train-logloss:0.448694  valid-logloss:0.457882
[170]   train-logloss:0.446672  valid-logloss:0.45617
[180]   train-logloss:0.444583  valid-logloss:0.454427
[190]   train-logloss:0.442908  valid-logloss:0.453044
[200]   train-logloss:0.441477  valid-logloss:0.451878
[210]   train-logloss:0.440077  valid-logloss:0.450777
[220]   train-logloss:0.438583  valid-logloss:0.449577
[230]   train-logloss:0.437216  valid-logloss:0.448519
[240]   train-logloss:0.435888  valid-logloss:0.447488
[250]   train-logloss:0.434745  valid-logloss:0.446607
[260]   train-logloss:0.433445  valid-logloss:0.445622
[270]   train-logloss:0.432287  valid-logloss:0.444771
[280]   train-logloss:0.431125  valid-logloss:0.443923
[290]   train-logloss:0.430027  valid-logloss:0.443119
[300]   train-logloss:0.42905   valid-logloss:0.442392
[310]   train-logloss:0.428125  valid-logloss:0.441735
[320]   train-logloss:0.427162  valid-logloss:0.441049
[330]   train-logloss:0.426058  valid-logloss:0.440266
[340]   train-logloss:0.425016  valid-logloss:0.439559
[350]   train-logloss:0.424074  valid-logloss:0.438942
[360]   train-logloss:0.423197  valid-logloss:0.438411
[370]   train-logloss:0.422121  valid-logloss:0.437743
[380]   train-logloss:0.421351  valid-logloss:0.437298
[390]   train-logloss:0.42033   valid-logloss:0.436687
[399]   train-logloss:0.41964   valid-logloss:0.436289
```

测试集：



对预测后结果进行处理后：

```
def adj(x,te=0.173,tr=0.369):
    a=te/tr
    b=(1-te)/(1-tr)
    return a*x/(a*x+b*(1-x))
res = sub.copy()
res.is_duplicate=res.is_duplicate.apply(adj)
res.to_csv('xgboost_sub_adj.csv',index=False)
```



5 项目结论

5.1 结果

最后可以看出 XGBT 得到了最好的效果，未加主题模型等特征时分数为 0.41964，现在提高到 0.38355,能最好的分辨两个句子之间的相似度。

5.2 对项目的思考

使用超参数需要不断的测试，有时候合理的参数搭配需要对于算法有一个比较细节的了解，知道超参数的大致范围，然后通过 gridsearch 或者别的 tuning 包找到最佳参数。

5.3 需要作出的改进

测试集数据集非常大，会爆内存，我另外购置了内存条解决这个问题，同时可以探究虚拟内存设置，或者可以把测试集切换，分批次的进行特征工程和预测。

关于预测的后处理，测试集和训练集的分布是不一致的，我使用了 logloss 进行测评，通过评审给出的函数对预测的概率进行后处理。

尝试提取更多有效的特征，尝试组合不同的算法，使用神经网络继续探索。

[1] https://zh.wikipedia.org/wiki/%E8%AF%8D%E8%A2%8B%E6%A8%A1%E5%9E%8B

[2] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[3] https://zh.wikipedia.org/zh-tw/%E4%B8%BB%E6%88%90%E5%88%86%E5%88%86%E6%9E%90

[4] https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

[5] https://en.wikipedia.org/wiki/Word2vec

[6] Jeffrey Pennington, Richard Socher, Christopher D. Manning; GloVe: Global Vectors for Word Representation.

[7] https://ai.tencent.com/ailab/nlp/embedding.html

[8] Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM), 2333- 2338.

[9] Lu, Z., and Li, H. 2013. A Deep Architecture for Matching Short Texts. In Advances in Neural Information Processing Systems (NIPS), 1367-1375.

[10] Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM), 101-110.

[11] Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In Advances in Neural Information Processing Systems (NIPS), 2042-2050.

[12] Qiu, X., and Huang, X. 2015. Convolutional Neural Tensor Network Architecture for Community-Based Question Answering. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), 1305-1311.

[13] Palangi, H.; Deng, L.; Shen, Y.; Gao, J.; He, X.; Chen, J.; Song, X.; and Ward, R. K. 2015. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. CoRR abs/1502.06922.

[14] Yin, W., and Schutze, H. 2015a. Convolutional Neural Network for Paraphrase Identifica- tion. In The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL) , 901－911.

[15] Quora Question Pairs－Can you identify question pairs that have the same intent?
https://www.kaggle.com/c/quora-question-pairs

[16] https://en.wikipedia.org/wiki/Cross_entropy

[17] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova; BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL].

[18]
https://github.com/abhishekkrthakur/is_that_a_duplicate_quora_question/blob/master/feature_engineering.py