TAIL CAMP NLP 第一周项目报告 - 句子语义相似度预测

Contents

1. 项目简介

本次的项目目标是"预测教师培训或类似场景中内容主题的相似性, 提高教学效率".

实战训练阶段, 我们在一个简化的场景下进行学习: 给定一系列的英文句子对, 每个句子对中的两个句子, 在语义上具有一定的相似性, 用 0 到 5 之间的分值来衡量两个句子的语义相似性. 语义相似性越高, 得分越高.

示例如下:

	id	sent1	sent2	similarity
0	10001	two big brown dogs running through the snow.	A brown dog running through the grass.	2.000
1	10002	A woman is peeling a potato.	A woman is slicing a tomato.	1.333
2	10003	A small boy is building with some wooden blocks.	A small boy wearing a blue shirt stacks wooden...	4.000

可以出, 对两个句子的相似性进行评分是一项很主观的任务. 主观性带来的结果是很难有一个统一的评价指标, 这使得任务更加艰巨.

用机器学习的方法取代或辅助人类对相似性评分, 能极大地弥补人工评分的不足, 同时提高效率.

上述 `df_train` 就是项目的训练数据, `df_test`则是测试数据. 训练的目标是通过对训练数据的学习, 使得机器习得人类对句子语义相似性的评估.

2. 思路与实现

2.1 数据分析

在我们看来, 本项目属于 Text Mining 的范畴. 磨刀不误砍柴工, 首先对数据进行分析.

> 助教已经提供了一份较为完善的 Text Analysis, 为保证报告完整性, 我们选择了加上 Text Analysis 这一部分.

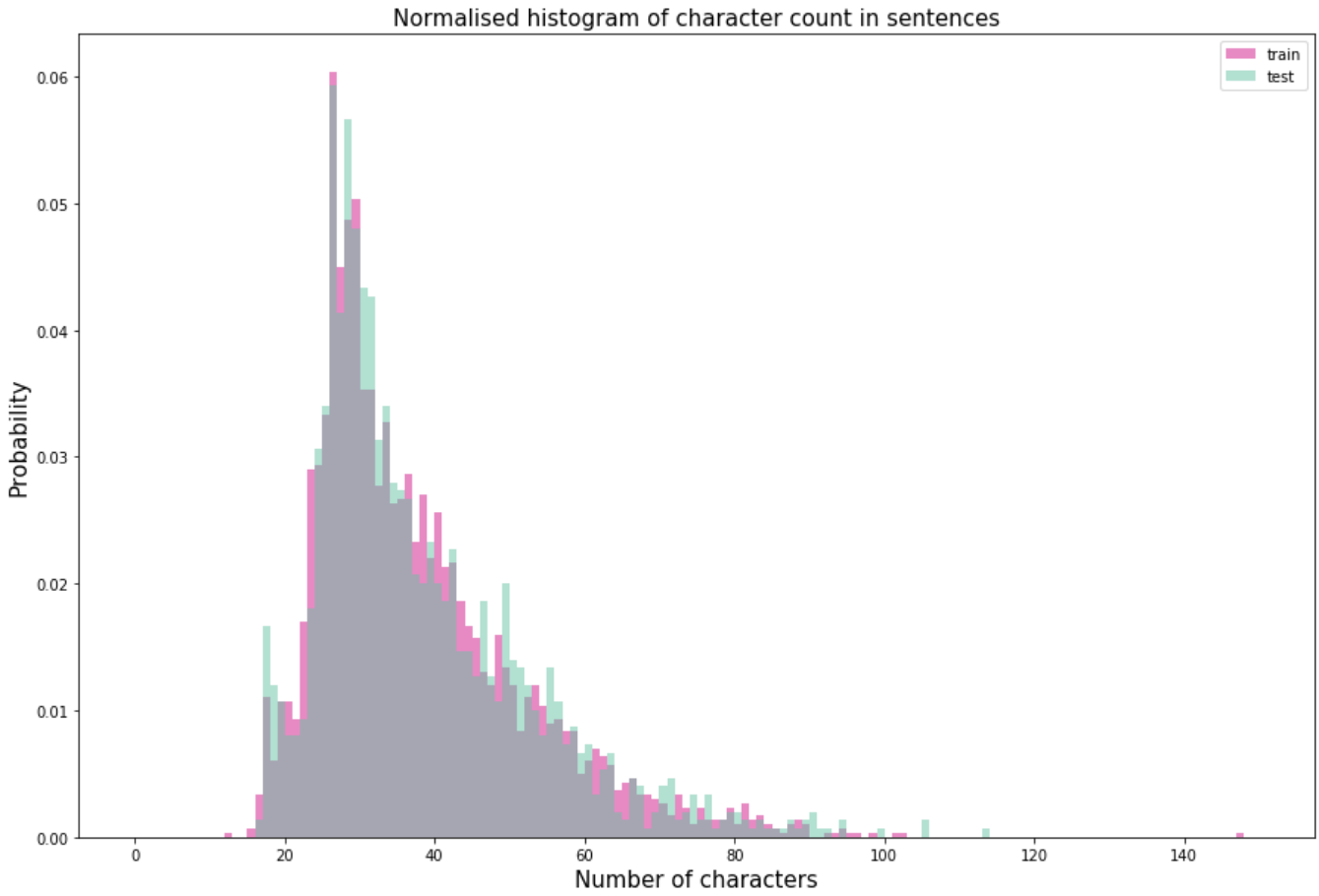
首先, 我们对训练数据与测试数据的基本信息进行对比, 简单地考虑以下两方面:

- 句子长度;
- 句子包含的单词数 (包括停词)

训练集 vs. 测试集 - 句子长度

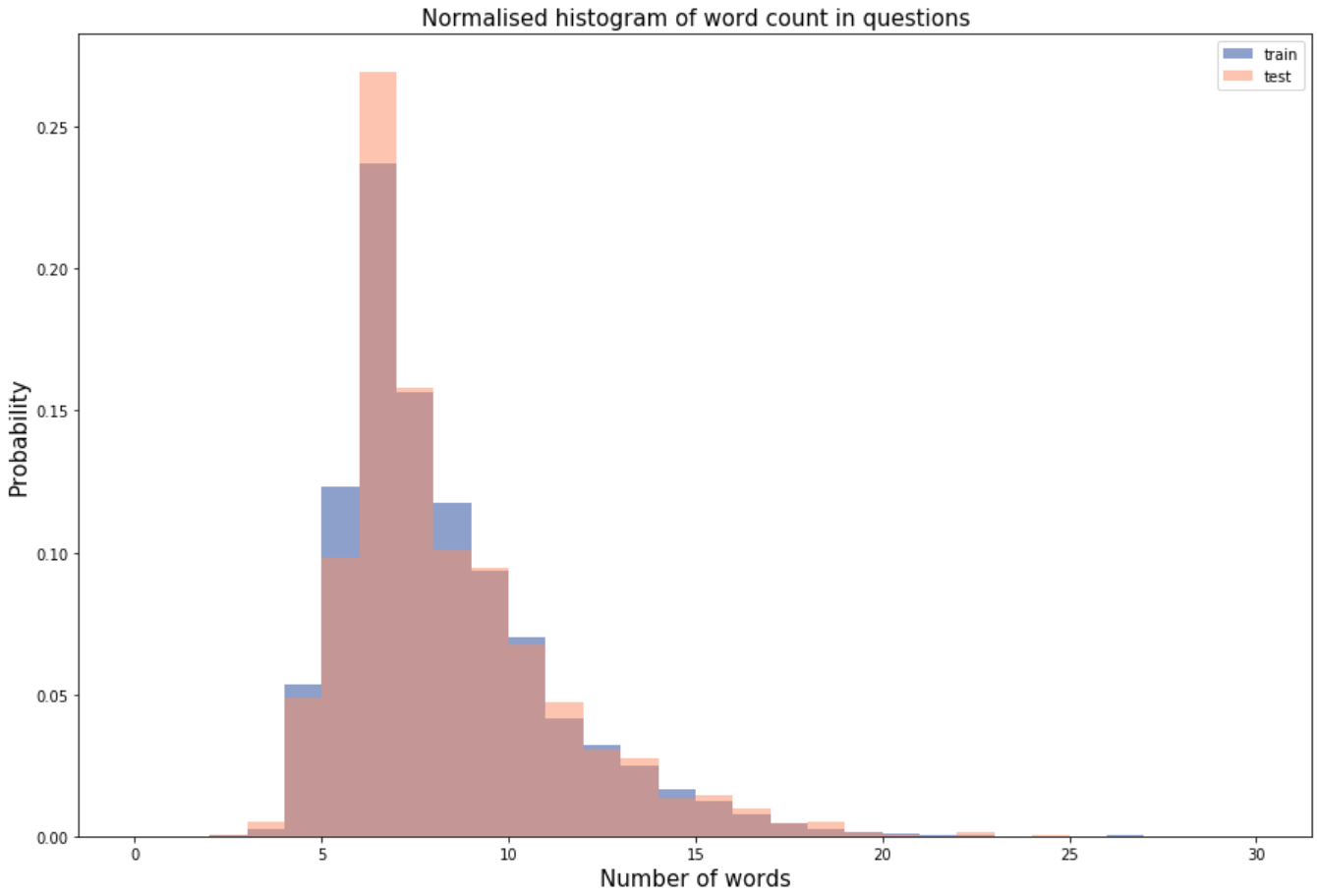
可以看到, 训练数据与测试数据在句子长度这一特征上, 概率分布基本吻合.

由于数据量较小, 直方图显得不那么平滑, 某些长度的句子在训练集和测试集中出现的概率有明显差异.



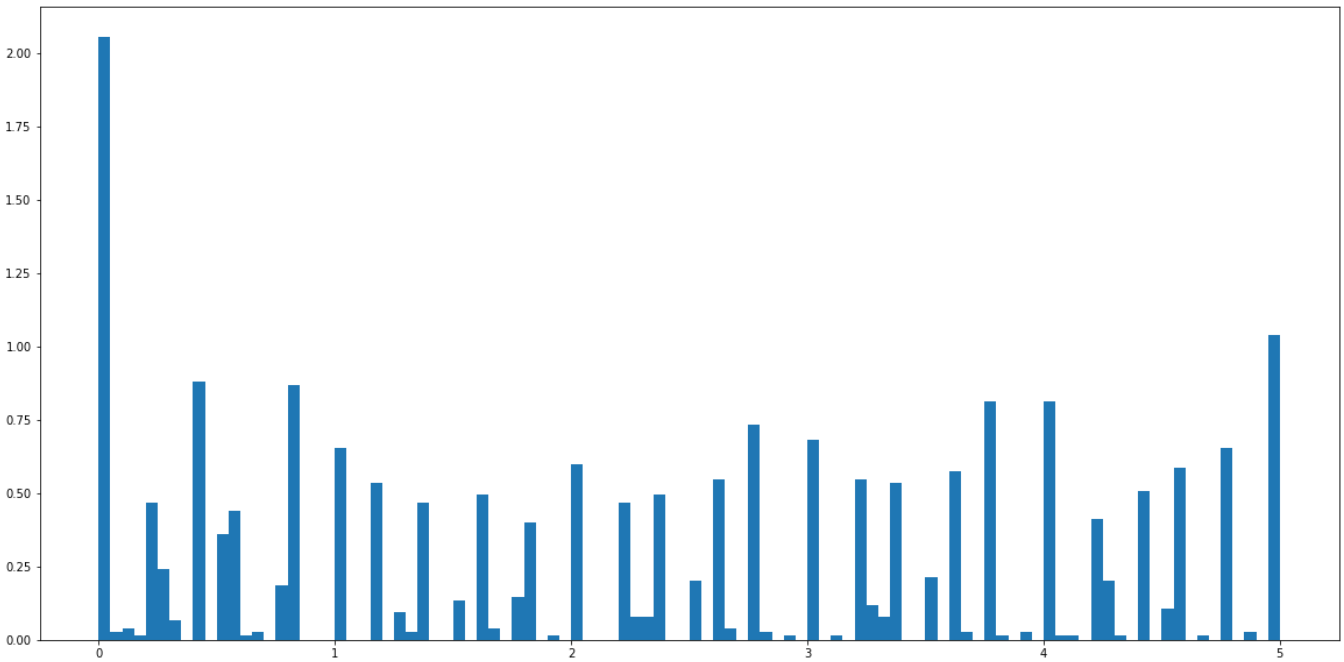
训练集 vs. 测试集 - 单词数

从下图可以看到, 训练数据与测试数据在单词数这一特征上, 概率分布也基本吻合, 且分布的形状与句子长度相似.



接下来我们对训练集中句子的语义相似性进行分析.

从下图的相似度分布中可以看出, 除了两个句子完全不具有相似性 (0 分) 的情况特别多之外, 句子的语义相似性在完全不相似与完全相同之间总体表现为均匀分布.



我们简单的数据分析到这里就结束了.

经过对训练数据与测试数据中的句子长度和单词数的对比, 我们基本可以确定, 训练数据与测试数据是独立同分步的.

因此, 我们可以有一个先验的印象: ***测试数据的相似性从完全不相似到完全相同之间大致也表现为均匀分布, 并且完全不相似的情况会更多.***

这给了我们两点启示:

- 在训练阶段, 可以首先对相似度为 0 的样本进行降采样;
- 在预测阶段, 可以将*预测值总体是否服从均匀分布*作为衡量预测结果的一项指标.

2.2 28 个特征 + XGBRegressor 的第一次尝试

Kaggle 上有一个Quora Question Pairs (https://www.kaggle.com/c/quora-question-pairs/overview)的竞赛, 其目标是判断两个提问是否重复了. 这和我们的任务具有一定的相似性.

因此, 我们首先在 Kaggle 的讨论区浏览了一些解决方案. 借鉴本文 (https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur/), 我们首先提取了 28 个特征, 包括:

- 字面特征 f1s:
  - 句子1 的长度
  - 句子2 的长度
  - 句子间的长度差
  - 句子1 的非空格字符数
  - 句子2 的非空格字符数
  - 句子1 的单词数
  - 句子2 的单词数
  - 句子间的共同单词比
- 模糊特征 f2s:
  - Q Ratio
  - W Ratio
  - Partial ratio
  - Partial token set ratio
  - Partial token sort ratio
  - Token set ratio
  - Token sort ratio
- 词向量相关的距离特征 f3s:
  - Word mover distance (第一次实验未包含)
  - Normalized word mover distance (第一次实验未包含)
  - 句子向量 (sentence vectors) 间的 Cosine 距离
  - 句子向量间的 Manhattan 距离
  - 句子向量间的 Jaccard 距离
  - 句子向量间的 Canberra 距离
  - 句子向量间的 Euclidean 距离
  - 句子向量间的 Minkowski 距离
  - 句子向量间的 Braycurtis 距离
  - 句子1 向量的 Skew 统计量
  - 句子2 向量的 Skew 统计量
  - 句子1 向量的 Kurtosis 统计量
  - 句子2 向量的 Kurtosis 统计量

一开始我们只使用了 XGBRegressor (`learning_rate=0.02`, `max_depth=4`), 在使用 Glove.6B.100d.txt (http://nlp.stanford.edu/data/wordvecs/glove.6B.zip) 的情况下, 在平台上的得分达到了 0.773975! 这在现在看来也是一个中等偏好的成绩. **结果与讨论**放在本文的第三部分, 在此先按下不表.

在项目训练阶段, 我们并没有考虑特征的有效性, 而是一股脑儿地全部"喂"给了模型.

在此, 我们对个别特征的有效性做一个简单的调研, 主要包括:

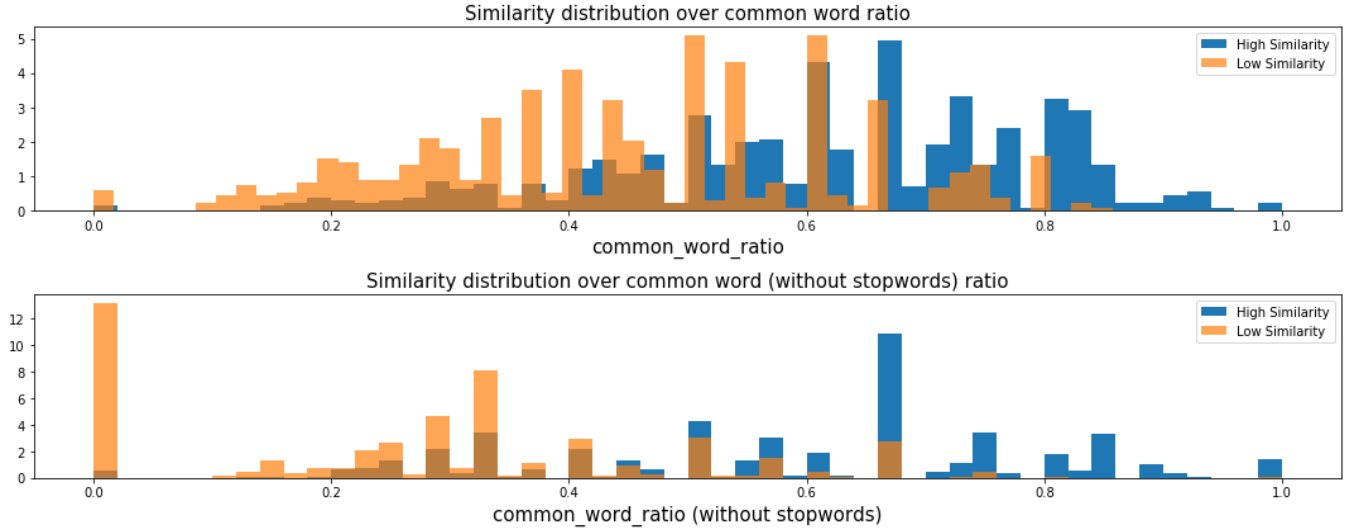
- 句子间共用单词数
- Q Ratio
- 句子向量间的 Cosine 距离

句子间共用单词比

我们首先调查了共用单词比与句子语义相似性的关系.

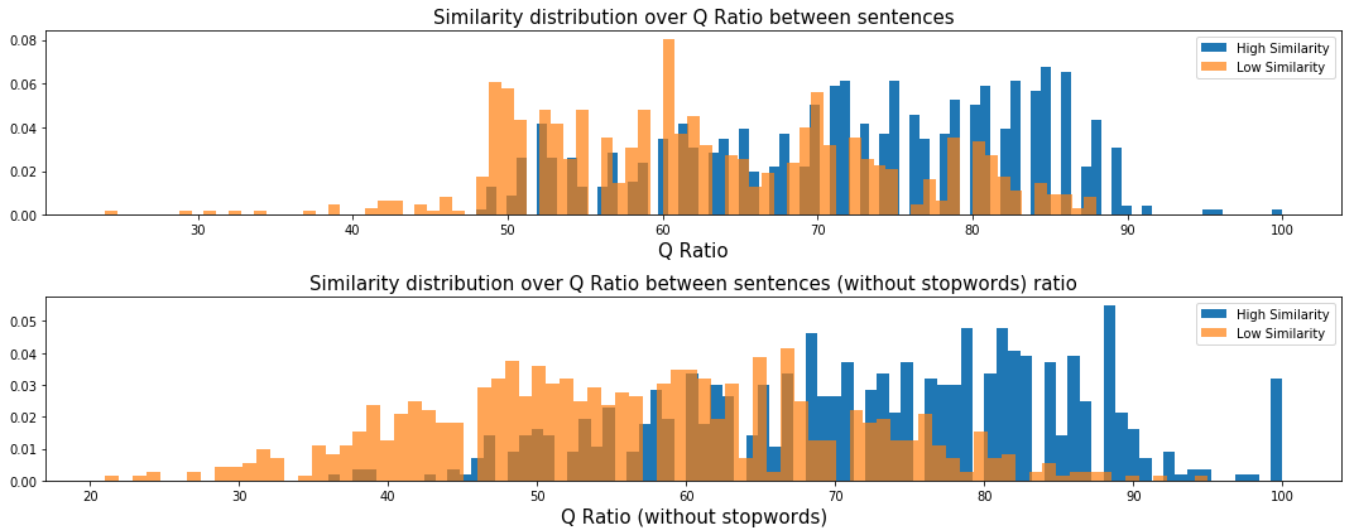
> 因为 similarity 的分布总体呈现为均匀分布, 我们就简单地以 2.5 为界, 将句子对分类为*相似*与*不相似*两类来考量特征的有效性.

我们发现, 就这一维度而言, 语义相似性的分布有较明显的差异, 表明这是一个相对有力的特征. 并且, 去掉停词之后, 这种差异更加明显.



句子间的 Q Ratio

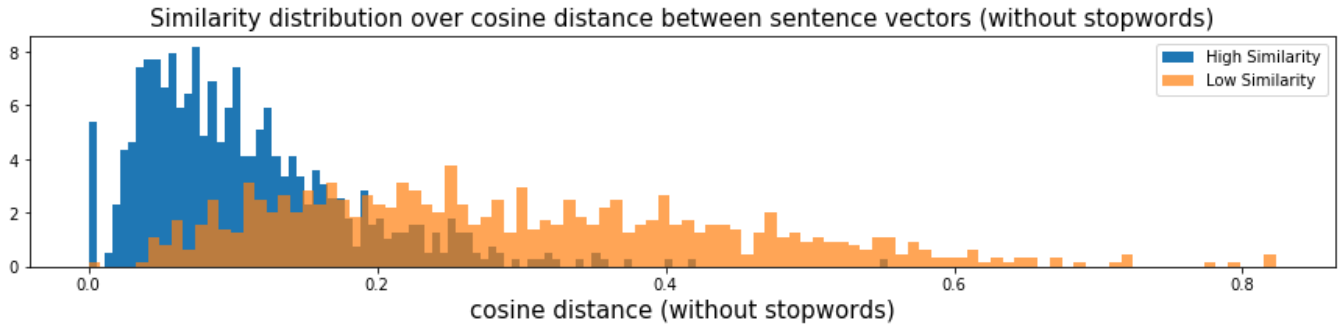
同共用单词比一样, 语义相似性在 Q Ratio 上分布的差异较为明显, 且去掉停词之后, 差异表现得更为明显. 这表明 Q Ratio 同样是一个可用的特征.



句子向量间的 cosine distance

如下图所示, (去掉停词的)相似句子向量间的 cosine distance 普遍偏小, 且分布相对集中; 而不相似句子向量间的 cosine distance 偏大, 且分布跨度很宽.

因此句子向量间的 `cosine distance`也是一个有效的特征.

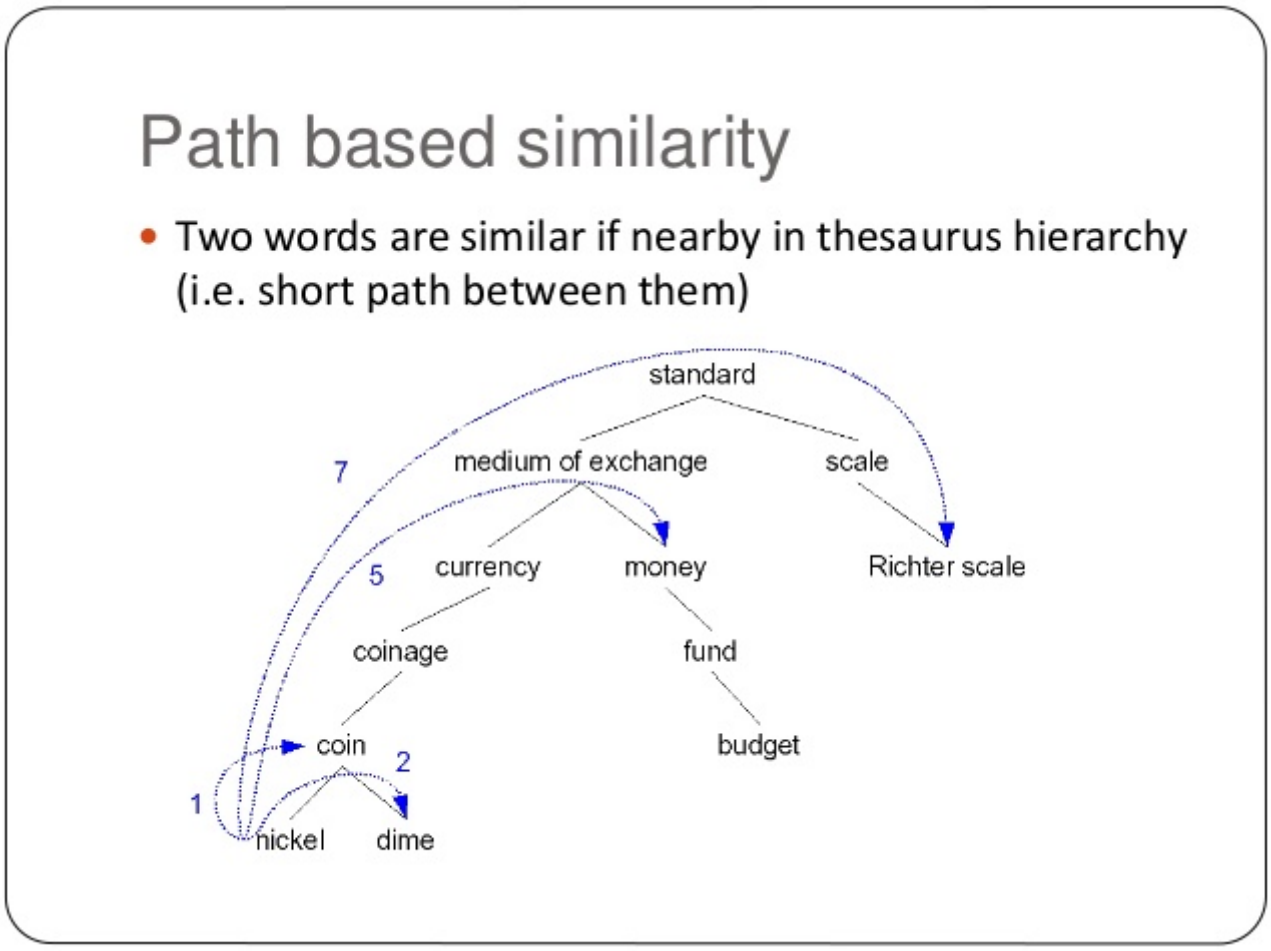


2.3 WordNet-based similarity

在了解了本周的项目之后, 我们首先上网搜索了相关资料. 恰好 Coursera 上有一门讲 Python Text Mining (https://www.coursera.org/learn/python-text-mining/home/info) 的课程, 并且在第四周的课上介绍了 Semantic Text Similarity (基于 WordNet) 。

下图是基于 WordNet 计算 Semantic Text Similarity 的一个示例. 简单地说, WordNet 是一个英语词典, 它以树的形式将同义词组织起来. (就像界门纲目科属种的生物分类法).

其中一种度量单词相似性的方法称为`Path Similarity`, 距离的表示见下图.



在本项目中, 我们采用了两种不需要句子之外的数据 (比如辅助字典) 的两种相似度度量方法, Path Similarity 和 Wup Similarity.

以此求句子相似性的思路如下:

1. 从句子中提取单词, 并且求每个单词的同义词集 (Synsets);
2. 然后对句子1 中的每个单词, 计算它的与句子2 中所有单词的 Path/Wup Similarity, 取其中的最大值, 对句子1 中所有求得的这样的最大值求均值作为句子1 对于句子2 的相似度;
3. 由于这是一个非对称的相似性, 因此再以同样方法求句子2 对于句子1 的相似度;
4. 以两个相似度的均值作为最终句子的相似度.

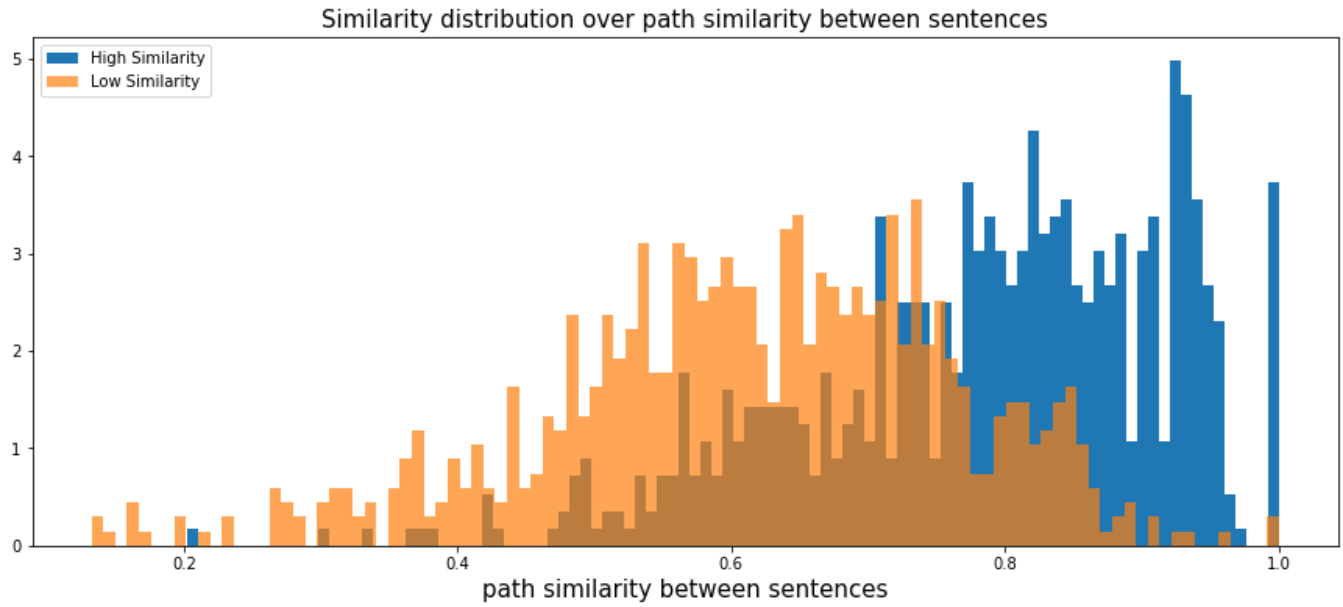
这样, 我们又得到 2 个特征:

- 基于 WordNet 的相似度特征 f4s:
  - Path Similarity;
  - Wup Similarity

同样地, 我们对`Path Similarity`的有效性进行调查.

可以看到, 相似句子间的 Path Similarity 普遍高于 0.5, 集中在 0.8; 而不相似句子的 Path Similarity 的分布较为分散, 集中在 0.6.

因此将 Path Similarity 作为特征是可行的.



事实上, 增加了 Path Similarity 和 Wup Similarity 两个特征之后, 成绩提升了 0.01 (0.813395 -> 0.824451)

2.4 更多特征!

在看到**特征带来的红利**之后, 我们尝试获得更多特征, 增加了如下特征:

- gensim.doc2vec.docvecs.similarity f5s
- 基于 tf-idf 的距离特征 f6s:
  - cosine distance
  - manhattan distance
  - euclidean_distance
- 基于 lsa 向量的距离特征 f7s:
  - cosine distance
  - manhattan distance
  - jaccard distance
  - canberra distance
  - euclidean distance
  - minkowski distance
  - braycurtis distance
- 基于 lda 向量的距离特征 f8s (此类特征是有害或者无效的, 实际未使用, 此后不再讨论):
  - cosine distance
  - manhattan distance
  - jaccard distance
  - canberra distance
  - euclidean distance
  - minkowski distance
  - braycurtis distance
- 编辑距离相关的特征 f9s (此类特征也是有害的或无效的, 实际并未使用, 此后不再讨论):
  - levenshtein distance
  - jaro distance
  - jaro-winkler distance
  - ratio similarity

所有这些特征, 包括之前未加进来的 Word Mover Distance 以及 Normalized Word Mover Distance, 在单一 XGBRegressor 和 Glove.840B.300d.txt (http://nlp.stanford.edu/data/wordvecs/glove.840B.300d.zip) 的情况下, 取得了 0.849993 的成绩!

而使用粗糙的 Ensemble 方法, 结合未经过参数选择的 SVR, RandomForestRegressor, GradientBoostingRegressor, LightGBMRegressor, XGBRegressor, 成绩达到了 0.851685!

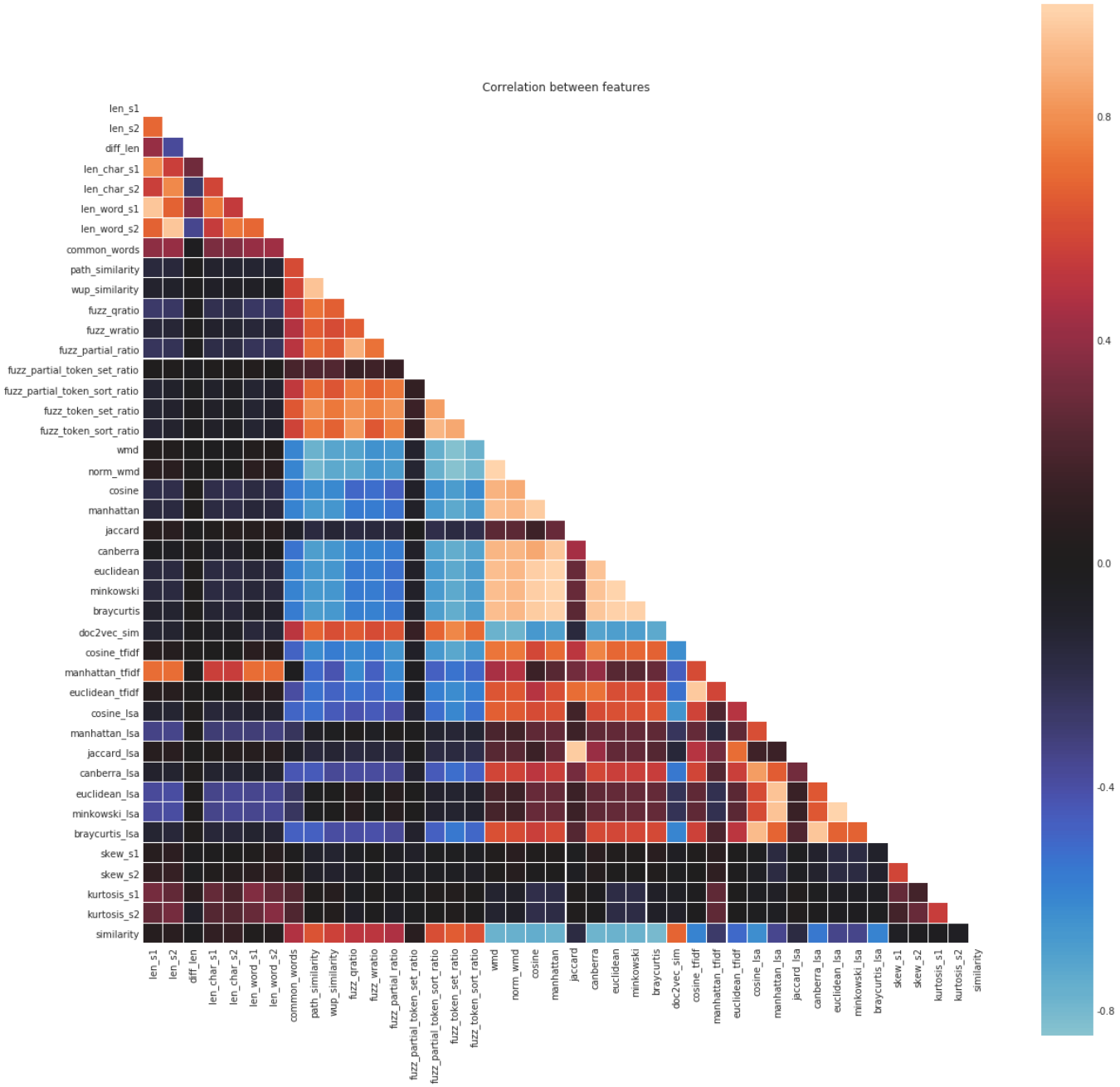
由于我们使用了比较多的特征, 无法一一讨论它们与相似度的相关程度. 因此, 我们仅从相关性上去一探特征的有效性.

如下图所示, 最后一行是句子语义相似性与各特征的相关性. 我们需要与相似性强正相关或强负相关的特征, 而那些与相似性并不相关的, 在训练模型的过程中可以忽略.

可以看到, 上述的共用单词比, `Q Ratio`, 句子向量的 `cosine distance`, `path similarity`都是与相似性具有强正/负相关性的. (事先并没有经过筛选, 很意外地, 它们都被选中了)

此外, 表示相似性的特征与句子语义相似性成正相关关系, 而表示距离的特征与句子语义相似性成负相关关系. 这符合基本逻辑: ***距离越远, 越不相似.***

(图中还展示了特征间的相关性, 当然这并不是我们研究的重点, 不再展开讨论)



3. 结果与讨论

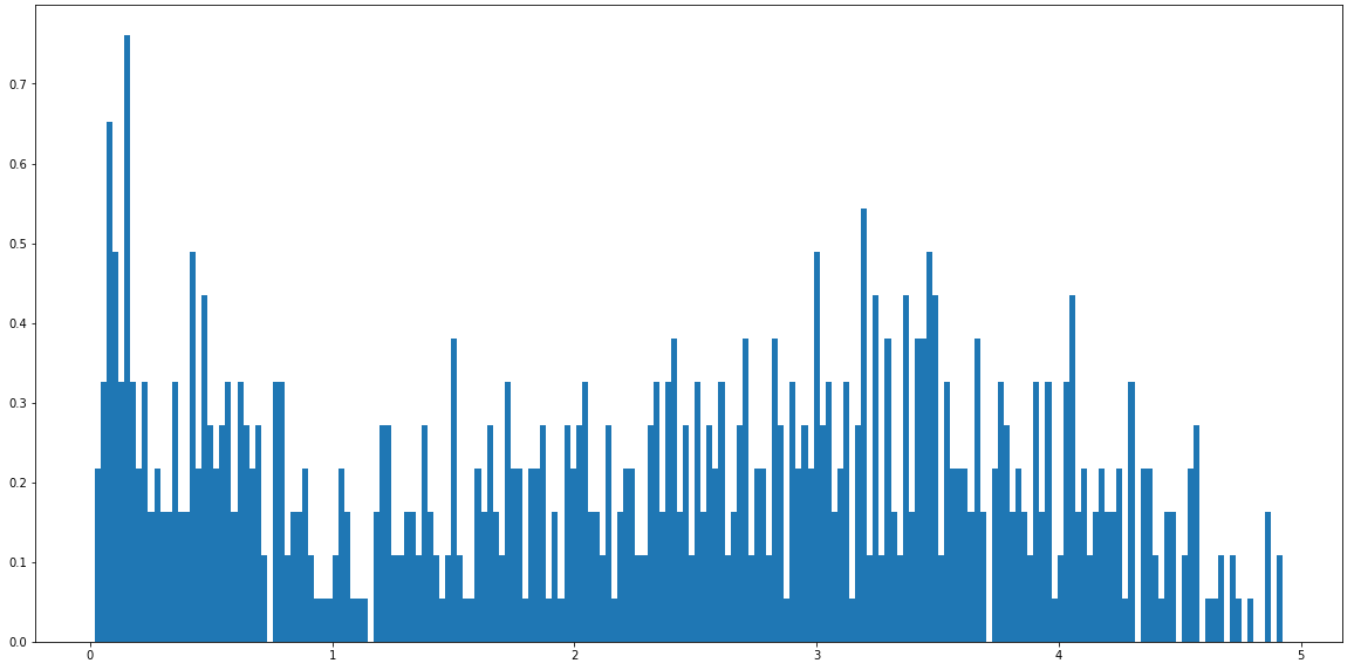
3.1 预测值的概率分布

上一节, 我们提出了一个先验假设, 即:

> 训练数据与测试数据独立同分布, 那么测试数据的相似性预测值总体也应该服从均匀分布

看看我们的预测结果是否如此. 与最终提交的项目保持一致, 此处同样使用了 ensemble 的方法. 得到的预测值分布如下图所示.

可以看到, 不相似的比例确实更大一些, 概率分布也挺像均匀分布的, 只是相似度在 3 到 4 之间的预测值似乎偏多了. 也许这正是我们的模型的不足之处吧, 没有拟合好这部分的数据.


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5

3.2 更多特征与更好模型带来的性能提升

本项目, 我们在特征提取上投入了更多的关注度, 而模型只是从 XGBRegressor 提升到了 SVR, RandomForestRegressor, GradientBoostingRegressor, LightGBMRegressor, XGBRegressor 的 Ensemble. (说是　Ensemble 其实也不太对, 我们只是对各 Regressors 的结果求了平均. 姑且称为 Ensemble 吧)

我们记录了使用不同特征与模型的成绩, 不妨看一下它们带来的性能提升.

> f[n]s 表示的特征见上文各特征出处.

模型	预训练的词向量	特征	成绩	注释
XGBRegressor	Glove 100d	f1s + f2s + f3s	0.773975	不含 Word Mover Distance
XGBRegressor	Word2Vec 300d	f1s + f2s + f3s	0.797716	不含 Word Mover Distance
XGBRegressor	Fasttext 300d	f1s + f2s + f3s	0.811882	不含 Word Mover Distance
XGBRegressor	Glove 300d	f1s + f2s + f3s + f4s	0.824451	不含 Word Mover Distance
粗糙的 Ensemble	Glove 300d	f1s + f2s + f3s + f4s	0.842447	-
XGBRegressor	Glove 300d	f1s + f2s + f3s + f4s + f5s + f6s	0.845027	-
粗糙的 Ensemble	Glove 300d	f1s + f2s + f3s + f4s + f5s + f6s	0.851685	-
XGBRegressor	Glove 300d	f1s + f2s + f3s + f4s + f5s + f6s + f7s	0.849993	-
微调的 Ensemble	Glove 300d	f1s + f2s + f3s + f4s + f5s + f6s + f7s	0.855244	-

从结果来看, 更多特征带来的性能提升更明显. 但在我们的实践中, f8s 和 f9s 并不能再带来性能的提升, 反而使性能略微下降了.

Ensemble 带来的性能提升有限, 我们认为有 3 方面原因:

1. 使用方法不对, 或者各 Regressor 并调至最优参数状态;
2. XGBRegressor 太强了;
3. 太迟引入 Ensemble, 此时即使增加更多特征, 带来的性能提升也开始下降, 因此并不能归因于 Ensemble 不好.

4. 回顾与总结

4.1 陈同学版

我是深度学习派的, 一开始我先了解了学习资料上提供的将深度学习应用到 STS 问题的方法, 跃跃欲试. 周二和周三, 我一直尝试用深度学习的方法. 我"借用"了两个模型:

1. Decomposable attention model https://arxiv.org/abs/1606.01933 (https://arxiv.org/abs/1606.01933)
2. ESIM https://arxiv.org/abs/1609.06038 (https://arxiv.org/abs/1609.06038)

但是, 效果都不好, 连差强人意都算不上, 最好的成绩是0.3+. 此外, 使用预训练好的词向量, 反而使结果更更更差了!

我猜测，可能有以下一些原因**:**

1. 本项目的数据量实在太小, 撑不起深度模型. 对于深度学习, 一般来说, 数据越多越好; (用牛刀杀鸡其实也不好杀)(欠拟合)
2. 只利用了词向量这一项特征, 没有充分利用其余特征;
3. 以上两个模型比较复杂, 对于我们的数据量, 马上就过拟合了. (至于是欠拟合还是过拟合, 效果不好, 我就没去研究了)

（还望不吝赐教）

周四, 我无奈地转向了并不熟悉的传统机器学习方法, 再也没有回过头看一眼深度学习. 现在回过头来看, 后面特征多了, 如果再结合深度学习的话, 结果应该会更好. 比如为了适配传统方法, 我抛弃了所有的向量特征, 而它们可能是极有价值的, 比如 Doc2Vec 模型得到的句子向量.

如前文所述, 由于看到**增加特征**带来的巨大性能提升, 后面我就专注于增加更多特征, 而没有对已有的特征进行分析 (本文的相关性分析表明, 部分特征是无效的), 也没有探索更好的模型.

因此, 本项目带给我最大的启迪是:

1. 有效的特征带来的帮助真的很大, 多多益善?!
2. 深度学习并非万能的! 深度学习有所短, 传统方法亦有所长. (并非指深度学习不适用于 STS, 而是各方面原因加起来, 我觉得本项目中, 深度学习只能当配角)
3. 本项目让我更加深刻地体会到没有免费的午餐这一条神则!

4.2 李同学版

首先，感谢助教大佬，共享了测试demo。我一脸懵逼时，仔细学习了一下他的代码。才知道处理 NLP 的主要步骤就是将其转换成词向量，然后应用模型。所以，baseline 就在他的 demo 的基础上运用了 Day6 里的 TFIDF 进行文本向量化，跑出来结果 0.76 还不错~

第二，感谢队友大腿，在我可怜巴巴只知道两个特征时，分享了他的语料和参考网址。才意识到要特征多一些，准确率才能上去。所以，开始各处扒拉特征。

第三，感谢群里小伙伴的热烈讨论。学习到解决这个问题的两个方案：神经网络和特征+传统模型。选择了后者，因为神经网络没怎么学过。最后选了 11 个特征 + 三个模型，得分上了 0.81。

第四，第一次接触NLP，第一次完成一个小项目，这种赶在 deadline 前完成任务的感觉很开心。但是因为自己学习的不够多，所以没什么经验分享给大家，下周一定努力学习。

不足之处：最后觉得堆特征太麻烦了，就弃疗了。后来意识到特征工程是项目实现的重要一步，要全面把握各个特征，才能取得好成绩。

5. 参考资料

参考资料

- Natural Language Processing with Python (http://www.nltk.org/book/)
- Coursera Python text mining - Semantic Text Similarity (https://www.coursera.org/learn/python-text-mining/lecture/DpNWl/semantic-text-similarity)
- Kaggle 第一名的解决方案 (https://www.kaggle.com/c/quora-question-pairs/discussion/34355)
- Lam Dang's Deep Model (https://www.kaggle.com/lamdang/dl-models/code)
- 数据分析以及 XGBoost Startup (https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb/comments)
- Is That a Duplicate Quora Question? (https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur/)
- scipy.spatial.distance (https://docs.scipy.org/doc/scipy/reference/spatial.distance.html)
- Scikit Learn - SVR (http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html)
- Scikit Learn - GridSearchCV (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)
- Scikit Learn - GradientBoostingRegressor (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html)
- Scikit Learn - RandomForestRegressor (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html)
- What is LightGBM, How to implement it? How to fine tune the parameters? (https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc)
- Scikit Learn - TfidfVectorizer (http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)
- XGBoost Python API (https://xgboost.readthedocs.io/en/latest/python/python_api.html)
- StackingRegressor (https://rasbt.github.io/mlxtend/user_guide/regressor/StackingRegressor/)
- Gensim Dov2Vec (https://radimrehurek.com/gensim/models/doc2vec.html)