目录

TAIL CAMP - NLP W1 3rd Solution 0.855

1 1.3 k 7

iamkissg LV2 2

Fork 点赞

内容 数据集 Fork记录 7 评论 1

内容

版本 2 2018-02-03 01:44

目
录

写在前面的话

> Kaggle 上面有一个Quora Question Pairs (https://www.kaggle.com/c/quora-question-pairs/overview)比赛，目标是判断两个问题是否重复了。和我们本周的学习任务非常相似。有兴趣的同学可以去观摩以下。

我一开始使用的是深度学习方法，使用了以下两个模型：

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)

用了这里 (https://www.kaggle.com/lamdang/dl-models/code)的模型（有修改），希望刷一个 Baseline 先。

但是，效果都不好，最好的成绩是0.3+。此外，使用预训练好的 Word Vector 的结果比在训练中学习 Word Vector 更差。这让人有些匪夷所思以。

我猜测，可能有以下一些原因：

1. 我们的数据量太小，而对于深度学习，一般来说，数据越多越好；（用牛刀杀鸡也不好杀）（欠拟合）
2. 只利用了 Word Vector，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。

（还望不吝赐教）

从周二到周四，坚持了2天深度学习方法，未果，我转向了传统方法，再也没有回过头看一眼深度学习。（后面如果再加上深度学习的话，结果可能会更好）

我的方法相对简单粗暴，就是提取尽可能多（有用）的特征，然后用 ensemble 的方法进行预测。

1. 特征

基于预训练的 Word Vector 的特征 f1s

- 句子间的 word mover distance 以及 normalized word mover distance（gensim 有相应方法）；
- 使用 word vector 粗略地求 sentence vector，再求各种距离，包括：
  - cosine distance
  - manhattan distance
  - jaccard distance
  - canberra distance
  - euclidean distance
  - minkowski distance
  - braycurtis distance
- 基于 sentence vector 的统计量，包括：
  - skew
  - kertosis

基于 Tfidf 等的特征

- 使用 gensim 的 Dov2Vec 模型求 sentence vector，自带 similarity 计算方法
- 基于 tfidf 的距离，包括：
  - cosine distance
  - manhattan distance
  - euclidean_distance
- 基于 lsa 的距离，包括：
  - cosine distance
  - manhattan distance
  - jaccard distance
  - canberra distance
  - euclidean distance
  - minkowski distance
  - braycurtis distance

字面特征 f2s

- 两个句子各自的长度以及长度差；
- 两个句子各自的字符数；
- 两个句子各自的单词数；
- 两个句子共用单词数。

模糊特征 f3s

此外使用了 fuzzywuzzy 一个神奇的库，提取的特征包括：

此处使用了 fuzzywuzzy 一个神奇的库，提取的特征包括：

- QRatio
- WRatio
- Partial Ratio
- Partial Token Set Ratio
- Partial Token Sort Ratio
- Token Set Ratio
- Token Sort Ratio

目
录

基于 NLTK WordNet 的特征 f4s

- Path Similarity
- Wup Similarity

计算了，但性能不再提升，反而下降的特征 f5s

- 基于 LDA 的各种距离（见上）
- 一些 edit distance 相关的距离，由 Python-Levenshtein 库提供，包括：
  - levenshtein
  - hamming（使用要求比较高，等长，没用）
  - jaro
  - jaro-winkler
  - ratio
  - sequence ratio

2. 模型

一开始我只使用了 XGBRegressor，最好的结果有 0.849993。（只使用 f1s + f3s + f4s + glove 300d，0.813）

使用 ensemble 的方法，加上 SVR、RandomFrorestRegressor、GradientBoostingRegreesor、LBGMRegressor，基本使用默认参数，对结果求平均，成绩大概在 0.853。

对各 Regressors 的参数进行简单挑选，成绩能达到 0.855。

另外我尝试了助教-常永炷提供的 Stacking 方法，应该是参数没设置好，性能反而下降了。

In [1]:

```
# 查看当前挂载的数据集目录
!ls /home/kesci/input/
```

```
Education_NLP  word2vec3861
```

In [2]:

```
# 查看个人持久化工作区文件
!ls /home/kesci/work/
```

```
kesci_submit  submission_sample
```

In [3]:

```
# 查看当前kernerl下的package
!pip list --format=columns
```

```
Package                  Version
------------------------ -----------
alabaster                0.7.10
altair                   1.2.0
arrow                    0.10.0
attrs                    17.3.0
Babel                    2.5.1
bayesian-optimization    0.5.0
beautifulsoup4           4.6.0
biopython                1.68
blaze                    0.10.1
bleach                   2.1.1
bokeh                    0.12.4
boto                     2.48.0
boto3                    1.4.8
botocore                 1.8.5
bs4                      0.0.1
bz2file                  0.98
CacheControl             0.12.3
click                    6.7
click-plugins            1.0.3
cligj                    0.4.0
colorlover               0.2.1
```

```
conda                     4.2.11
cufflinks                 0.8.2
cycler                    0.10.0
cytoolz                   0.8.2
datashape                 0.5.2
deap                      1.2.2
decorator                 4.1.2
Delorean                  0.6.0
docopt                    0.6.2
docutils                  0.14
dora                      0.1
entrypoints               0.2.3
ffm                       7e8621d
fitter                    1.0.8
Flask                     0.12.2
Flask-Cors                3.0.3
funcy                     1.10
future                    0.16.0
gensim                    2.2.0
Geohash                   1.0
gpxpy                     1.1.2
h5py                      2.6.0
haversine                 0.4.5
heamy                     0.0.7
hmmlearn                  0.2.0
html5lib                  1.0b10
humanize                  0.5.1
hyperopt                  0.1
imagesize                 0.7.1
ipykernel                 4.6.1
ipython                   6.2.1
ipython-genutils          0.2.0
ipywidgets                7.0.5
iso3166                   0.8
itsdangerous              0.24
jedi                      0.11.0
jieba                     0.38
Jinja2                    2.10
jmespath                  0.9.3
joblib                    0.11
jsonschema                2.6.0
jupyter                   1.0.0
jupyter-client            5.1.0
jupyter-console           5.2.0
jupyter-core              4.4.0
jupyter-kernel-gateway    1.2.0
jupyter-pip               0.3.1
Keras                     2.0.9
langid                    1.1.6
lightgbm                  2.0.5
line-profiler             2.0
mapbox                    0.14.0
MarkupSafe                1.0
matplotlib                2.1.0
matplotlib-venn           0.11.5
MDP                       3.5
milk                      0.6.1
missingno                 0.3.4
mistune                   0.8.1
ml-metrics                0.1.4
mlxtend                   0.9.1
mpld3                     0.3
mplleaflet                0.0.5
msgpack-python            0.4.8
multipledispatch          0.4.9
nbconvert                 5.3.1
nbformat                  4.4.0
netaddr                   0.7.19
networkx                  2.0
nltk                      3.2.2
nose                      1.3.7
notebook                  5.0.0
numexpr                   2.6.4
numpy                     1.13.3
odo                       0.5.0
olefile                   0.44
orderedmultidict          0.7.11
pandas                    0.19.2
pandas-profiling          1.4.0
pandocfilters             1.4.2
parso                     0.1.0
patsy                     0.4.1
pexpect                   4.3.0
```

```
pickleshare               0.7.4
Pillow                    4.0.0
pip                       9.0.1
plotly                    2.0.1
pluggy                    0.6.0
polyline                  1.3.2
prompt-toolkit            1.0.15
protobuf                  3.5.0.post1
psutil                    5.4.1
ptyprocess                0.5.2
pudb                      2017.1
py                        1.5.2
pyasn1                    0.4.2
pycosat                   0.6.1
pycrypto                  2.6.1
pydot                     1.2.3
pyecharts                 0.2.7
pygal                     2.4.0
Pygments                  2.2.0
pyLDAvis                  2.1.1
pymongo                   3.5.1
PyOpenGL                  3.1.0
pyparsing                 2.1.10
pytest                    3.3.0
python-dateutil           2.6.1
pytz                      2017.3
PyWavelets                0.5.2
PyYAML                    3.12
pyzmq                     16.0.3
qtconsole                 4.3.1
requests                  2.11.1
revrand                   0.6.5
rsa                       3.4.2
ruamel-yaml               -VERSION
s2sphere                  0.2.4
s3transfer                0.1.12
sacred                    0.6.10
scikit-image              0.13.1
scikit-learn              0.18.1
scipy                     0.18.1
seaborn                   0.7.1
setuptools                27.2.0
SexMachine                0.1.1
simplegeneric             0.8.1
six                       1.11.0
smart-open                1.5.3
smhasher                  0.150.1
snowballstemmer           1.2.1
Sphinx                    1.6.5
sphinxcontrib-websupport  1.0.1
SQLAlchemy                1.1.15
statsmodels               0.8.0
tables                    3.3.0
tensorflow                1.0.0
terminado                 0.8.1
testpath                  0.3.1
textblob                  0.11.1
tflearn                   0.2.1
Theano                    0.8.2
toolz                     0.8.2
torch                     0.2.0.post3
torchvision               0.1.9
tornado                   4.5.2
TPOT                      0.6.8
tqdm                      4.19.4
traitlets                 4.3.2
trueskill                 0.4.4
tzlocal                   1.4
update-checker            0.16
uritemplate               3.0.0
urwid                     1.3.1
vega                      0.5.0
vida                      0.3
visvis                    1.10.0
wcwidth                   0.1.7
webencodings              0.5.1
Werkzeug                  0.12.2
wheel                     0.29.0
widgetsnbextension        3.0.8
wrapt                     1.10.11
xgboost                   0.6a2
xlrd                      1.0.0
```

安装和更新了一些库, 保持远程与本地的一致, 避免不必要的麻烦

目
录

In [4]:

```
!pip install -i https://pypi.douban.com/simple -U fuzzywuzzy fuzzywuzzy[speedup] gensim pyemd lightgbm python-leve
```

```
Collecting fuzzywuzzy
  Downloading https://pypi.doubanio.com/packages/3b/36/be990a35c7e8ed9dc176c43b5699cd971cec0b6f9ef858843374171df4f2/
Collecting gensim
  Downloading https://pypi.doubanio.com/packages/1f/45/2c89934244e7f4b81a8b9aaea7153edb1b271e44840b9839566d33510424/
    100% |████████████████████████████████| 15.9MB 84kB/s
Collecting pyemd
  Downloading https://pypi.doubanio.com/packages/34/61/0f2803463c695bdde498e63a6300f7878829427ecdb9144b6306883ce7f9/
    100% |████████████████████████████████| 71kB 10.6MB/s
Collecting lightgbm
  Downloading https://pypi.doubanio.com/packages/1e/04/d7cad86b3a2895c2c10bbbf5c29cfcb82ba22e2927e74d182cdc391d9f4e/
    100% |████████████████████████████████| 624kB 2.1MB/s
Collecting python-levenshtein
  Downloading https://pypi.doubanio.com/packages/42/a9/d1785c85ebf9b7dfacd08938dd028209c34a0ea3b1bcdb895208bd40a67d/
    100% |████████████████████████████████| 51kB 9.9MB/s
Collecting scipy>=0.18.1 (from gensim)
  Downloading https://pypi.doubanio.com/packages/1a/83/6aed4f564f3f5d338fd3c642f33d5ded0fc577da5f9a7d85ed6ba23c5d51/
    100% |████████████████████████████████| 49.6MB 27kB/s
Requirement already up-to-date: numpy>=1.11.3 in /opt/conda/lib/python3.5/site-packages (from gensim)
Collecting smart-open>=1.2.1 (from gensim)
  Downloading https://pypi.doubanio.com/packages/fb/c9/fa4099bf0818edc38fcb04db7b905227d5bbffccfce71e90bc42e52d56e7/
Requirement already up-to-date: six>=1.5.0 in /opt/conda/lib/python3.5/site-packages (from gensim)
Collecting scikit-learn (from lightgbm)
  Downloading https://pypi.doubanio.com/packages/bf/53/8c9c950a3cfaec16069df196c0b76ab05b3d1f0527f6bb97a30f4dda5240/
    100% |████████████████████████████████| 12.2MB 110kB/s
Collecting setuptools (from python-levenshtein)
  Downloading https://pypi.doubanio.com/packages/bb/e0/ea620f9ecbaaaf3ebb288cdbbe8cc20b6a789c17e42d8916662b218e3349/
    100% |████████████████████████████████| 491kB 2.7MB/s
Requirement already up-to-date: boto>=2.32 in /opt/conda/lib/python3.5/site-packages (from smart-open>=1.2.1->gensim
Requirement already up-to-date: bz2file in /opt/conda/lib/python3.5/site-packages (from smart-open>=1.2.1->gensim)
Collecting requests (from smart-open>=1.2.1->gensim)
  Downloading https://pypi.doubanio.com/packages/49/df/50aa1999ab9bde74656c2919d9c0c085fd2b3775fd3eca826012bef76d8c/
    100% |████████████████████████████████| 92kB 10.2MB/s
Collecting boto3 (from smart-open>=1.2.1->gensim)
  Downloading https://pypi.doubanio.com/packages/4d/62/0c905760ea5c9bc22ed7e2b6544229ce983b8dcdcdd393508a39babfe3e0/
    100% |████████████████████████████████| 133kB 8.7MB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests->smart-open>=1.2.1->gensim)
  Downloading https://pypi.doubanio.com/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/
    100% |████████████████████████████████| 143kB 8.4MB/s
Collecting certifi>=2017.4.17 (from requests->smart-open>=1.2.1->gensim)
  Downloading https://pypi.doubanio.com/packages/29/9b/25ef61e948321296f029f53c9f67cc2b54e224db509eb67ce17e0df6044a/
    100% |████████████████████████████████| 337kB 3.8MB/s
Collecting idna<2.7,>=2.5 (from requests->smart-open>=1.2.1->gensim)
  Downloading https://pypi.doubanio.com/packages/27/cc/6dd9a3869f15c2edfab863b992838277279ce92663d334df9ecf5106f5c6/
    100% |████████████████████████████████| 61kB 10.5MB/s
Collecting urllib3<1.23,>=1.21.1 (from requests->smart-open>=1.2.1->gensim)
  Downloading https://pypi.doubanio.com/packages/63/cb/6965947c13a94236f6d4b8223e21beb4d576dc72e8130bd7880f600839b8/
    100% |████████████████████████████████| 133kB 8.4MB/s
Requirement already up-to-date: jmespath<1.0.0,>=0.7.1 in /opt/conda/lib/python3.5/site-packages (from boto3->smart-
Collecting botocore<1.9.0,>=1.8.15 (from boto3->smart-open>=1.2.1->gensim)
  Downloading https://pypi.doubanio.com/packages/6f/de/6d2a65a7541cf192ddf5f8ddb7412badc3f682890d40b306585836420ebc/
    100% |████████████████████████████████| 4.0MB 339kB/s
Requirement already up-to-date: s3transfer<0.2.0,>=0.1.10 in /opt/conda/lib/python3.5/site-packages (from boto3->sma
Requirement already up-to-date: docutils>=0.10 in /opt/conda/lib/python3.5/site-packages (from botocore<1.9.0,>=1.8.
Requirement already up-to-date: python-dateutil<3.0.0,>=2.1 in /opt/conda/lib/python3.5/site-packages (from botocore
Building wheels for collected packages: pyemd, python-levenshtein, smart-open
  Running setup.py bdist_wheel for pyemd ... - \ | / - \ done
  Stored in directory: /home/kesci/.cache/pip/wheels/7a/a4/21/0739da53783a6f1b3e6b9459302278ff001a80fa17e2a81450
  Running setup.py bdist_wheel for python-levenshtein ... - \ | / done
  Stored in directory: /home/kesci/.cache/pip/wheels/59/89/67/fe005088ee3f6112d25ee0cc7e11a61890c8f3bc024977a3a6
  Running setup.py bdist_wheel for smart-open ... - \ done
  Stored in directory: /home/kesci/.cache/pip/wheels/29/45/50/9a993d6922d19c2cb90c037cb183de26f1abaafa3fbc951f65
Successfully built pyemd python-levenshtein smart-open
Installing collected packages: fuzzywuzzy, scipy, chardet, certifi, idna, urllib3, requests, botocore, boto3, smart-
  Found existing installation: scipy 0.18.1
    Uninstalling scipy-0.18.1:
      Successfully uninstalled scipy-0.18.1
  Found existing installation: requests 2.11.1
    Uninstalling requests-2.11.1:
      Successfully uninstalled requests-2.11.1
  Found existing installation: botocore 1.8.5
    Uninstalling botocore-1.8.5:
      Successfully uninstalled botocore-1.8.5
  Found existing installation: boto3 1.4.8
    Uninstalling boto3-1.4.8:
      Successfully uninstalled boto3-1.4.8
```

```
  Found existing installation: smart-open 1.5.3
    Uninstalling smart-open-1.5.3:
      Successfully uninstalled smart-open-1.5.3
  Found existing installation: gensim 2.2.0
    Uninstalling gensim-2.2.0:
      Successfully uninstalled gensim-2.2.0
  Found existing installation: scikit-learn 0.18.1
    Uninstalling scikit-learn-0.18.1:
      Successfully uninstalled scikit-learn-0.18.1
  Found existing installation: lightgbm 2.0.5
    Uninstalling lightgbm-2.0.5:
      Successfully uninstalled lightgbm-2.0.5
  Found existing installation: setuptools 27.2.0
    Uninstalling setuptools-27.2.0:
      Successfully uninstalled setuptools-27.2.0
Successfully installed boto3-1.5.1 botocore-1.8.15 certifi-2017.11.5 chardet-3.0.4 fuzzywuzzy-0.16.0 gensim-3.2.0 id
Traceback (most recent call last):
  File "/opt/conda/bin/pip", line 11, in <module>
    sys.exit(main())
  File "/opt/conda/lib/python3.5/site-packages/pip/__init__.py", line 233, in main
    return command.main(cmd_args)
  File "/opt/conda/lib/python3.5/site-packages/pip/basecommand.py", line 252, in main
    pip_version_check(session)
  File "/opt/conda/lib/python3.5/site-packages/pip/utils/outdated.py", line 102, in pip_version_check
    installed_version = get_installed_version("pip")
  File "/opt/conda/lib/python3.5/site-packages/pip/utils/__init__.py", line 838, in get_installed_version
    working_set = pkg_resources.WorkingSet()
  File "/opt/conda/lib/python3.5/site-packages/pip/_vendor/pkg_resources/__init__.py", line 644, in __init__
    self.add_entry(entry)
  File "/opt/conda/lib/python3.5/site-packages/pip/_vendor/pkg_resources/__init__.py", line 700, in add_entry
    for dist in find_distributions(entry, True):
  File "/opt/conda/lib/python3.5/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1949, in find_eggs_in_z
    if metadata.has_metadata('PKG-INFO'):
  File "/opt/conda/lib/python3.5/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1463, in has_metadata
    return self.egg_info and self._has(self._fn(self.egg_info, name))
  File "/opt/conda/lib/python3.5/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1823, in _has
    return zip_path in self.zipinfo or zip_path in self._index()
  File "/opt/conda/lib/python3.5/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1703, in zipinfo
    return self._zip_manifests.load(self.loader.archive)
  File "/opt/conda/lib/python3.5/site-packages/pip/_vendor/pkg_resources/__init__.py", line 1643, in load
    mtime = os.stat(path).st_mtime
FileNotFoundError: [Errno 2] No such file or directory: '/opt/conda/lib/python3.5/site-packages/setuptools-27.2.0-py
```

In [5]:

```python
# 加载数据分析常用库
import random


import nltk
import gensim
import scipy.stats as stats
import Levenshtein
import pandas as pd
import numpy as np
from scipy.spatial import distance
import lightgbm as lgb
import xgboost as xgb


from fuzzywuzzy import fuzz
from lightgbm import LGBMRegressor
from nltk.corpus import wordnet as wn
from nltk.corpus import stopwords
from gensim.models import TfidfModel, LsiModel, LdaModel, KeyedVectors
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from gensim.corpora import Dictionary
from gensim.similarities import MatrixSimilarity
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import pairwise
from sklearn.decomposition import TruncatedSVD, LatentDirichletAllocation
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from keras.preprocessing.text import text_to_word_sequence
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from mlxtend.regressor import StackingRegressor
```

```
/opt/conda/lib/python3.5/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecat
  "This module will be removed in 0.20.", DeprecationWarning)
Using TensorFlow backend.
```

In [ ]:

```
nltk.download(["wordnet", "stopwords", "averaged_perceptron_tagger"])

stopwords_eng = stopwords.words("english")
```

目
录

Path Similarity & Wup Similarity

在了解了本周的任务之后, 我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, 其在第四周的课上介绍了 Semantic Text Similarity（基于 WordNet）。

In [7]:

```
# Path Similarity & Wup Similarity

def convert_tag(tag):
    """Convert the tag given by nltk.pos_tag to the tag used by wordnet.synsets"""

    tag_dict = {'N': 'n', 'J': 'a', 'R': 'r', 'V': 'v'}
    try:
        return tag_dict[tag[0]]
    except KeyError:
        return None


def doc_to_synsets(doc):

    tokens = text_to_word_sequence(doc)  # tokenize

    token_pos = nltk.pos_tag(tokens)  # pos_tag

    synsets = []
    for tk, pos in token_pos:
        sn = wn.synsets(tk, convert_tag(pos))
        if sn:
            synsets.append(sn[0])

    return synsets


def similarity_score(s1, s2, mode="path_similarity"):

    scores = []
    for sn in s1:
        if mode == "path_similarity":
            tmp_scores = [sn.path_similarity(other) for other in s2 if sn.path_similarity(other)]
        if mode == "wup_similarity":
            tmp_scores = [sn.wup_similarity(other) for other in s2 if sn.wup_similarity(other)]

        if tmp_scores:
            scores.append(max(tmp_scores))
    return sum(scores) / len(scores)


def document_similarity(doc1, doc2):

    synsets1 = doc_to_synsets(doc1)
    synsets2 = doc_to_synsets(doc2)
    return (similarity_score(synsets1, synsets2, "path_similarity") + similarity_score(synsets2, synsets1, "path_s
           (similarity_score(synsets1, synsets2, "wup_similarity") + similarity_score(synsets2, synsets1, "wup_sim
```

In [8]:

```
df_train = pd.read_csv("/home/kesci/input/Education_NLP/Tal_SenEvl_train_136KB.txt", delimiter="\t", header=None,
df_test = pd.read_csv("/home/kesci/input/Education_NLP/Tal_SenEvl_test_62KB.txt", delimiter="\t", header=None, nam

print("Shape of training data:", df_train.shape, "\n"
      "Shape of test data:", df_test.shape)
```

```
Shape of training data: (1500, 4)
Shape of test data: (750, 4)
```

In [9]:

```
# 构建句子集, 方便后续操作
sentences = pd.concat([df_train.loc[:, "sent1"], df_train.loc[:, "sent2"],
                       df_test.loc[:, "sent1"], df_test.loc[:, "sent2"]]).tolist()

# TODO: 要不要去掉停词?
word_sequences = [text_to_word_sequence(s) for s in sentences]
tagged_sents = [TaggedDocument(ws, [str(i)]) for i, ws in enumerate(word_sequences)]  # 用于 docv2vec
```

目
录

Doc2Vec

设置 Seed, 以便复现结果. 根据 Gensim 的文档介绍, 需要设置 workers=1

In [10]:

```
# https://radimrehurek.com/gensim/models/doc2vec.html
doc2vec = Doc2Vec(tagged_sents, size=300, dm=0, alpha=0.25, min_alpha=1e-4, iter=30, workers=1, seed=609)

print(doc2vec.docvecs.similarity(33, 1533))
print(doc2vec.docvecs.similarity(1, 1501))
print(doc2vec.docvecs.similarity(0, 1500))
print(doc2vec.docvecs.similarity(2, 1502))
print(doc2vec.docvecs.similarity(4, 1504))
```

```
0.917491304106
0.607555024424
0.677264673297
0.694751910045
0.371273336797
```

Tf-Idf

In [11]:

```
tfidfv = TfidfVectorizer(ngram_range=(1, 4), stop_words=stopwords_eng)

tfidf_matrix = tfidfv.fit_transform(sentences)
```

LSI

In [12]:

```
svd = TruncatedSVD(n_components=100, n_iter=10, random_state=609)

lsa_matrix = svd.fit_transform(tfidf_matrix)

print(lsa_matrix[0])
```

```
[  3.47404409e-03   2.22373541e-02   1.72838771e-01   5.58169807e-02
  -2.19213110e-02  -8.18178483e-03   3.66040249e-02   1.30470387e-01
  -3.99864857e-02  -1.01993573e-01   6.17471152e-04  -6.21242873e-03
  -3.33902837e-02   5.48510466e-02  -1.37874640e-03  -5.22134813e-03
  -1.26079919e-02   1.56455573e-02  -3.90574864e-02   1.45242487e-02
  -2.43051121e-03  -3.74701694e-02  -3.47839700e-02   6.81345257e-03
   4.74136187e-03  -1.25011529e-02  -2.27929280e-02  -7.52336968e-03
  -6.67043999e-03  -2.00939700e-02  -1.83548647e-02   3.01742020e-02
   2.90576264e-03  -1.14253930e-01  -5.73091329e-02   5.74733109e-03
  -3.57797384e-02   4.23303307e-02  -2.60934898e-02   1.06558202e-02
   1.45942889e-02   3.22726453e-02   2.55122526e-03   6.17196916e-02
   1.65754359e-02  -6.06156029e-02   9.24780217e-03  -5.75012920e-02
  -1.29428654e-02  -7.74230902e-02   4.58985180e-02  -1.50671917e-02
   5.90622178e-02   1.58968088e-02  -2.06261244e-02  -8.81951799e-02
   1.51172136e-03  -8.03586340e-03  -4.61403742e-03  -2.25831095e-04
  -5.99561115e-03   4.74755907e-02  -9.91554705e-03   5.36315052e-02
  -4.29314729e-02  -6.54988080e-02   2.99452640e-03  -2.72854115e-02
  -2.05697325e-02  -2.66056485e-02  -1.71327609e-02  -2.15907976e-02
   2.71009931e-02   9.18967297e-03   2.79190015e-02   4.30269815e-03
   6.65410174e-02  -1.22513409e-02  -5.17106299e-02   5.79589831e-02
  -3.79155943e-02  -2.10338556e-02   2.77810762e-02  -6.23197879e-03
  -2.42363509e-03   1.12752667e-02  -1.19637978e-02  -2.33514236e-02
   1.28728120e-02  -3.83569797e-02  -3.03667619e-02   1.22272578e-02
   1.03565528e-02   1.00102709e-04   1.34599249e-03  -1.57614932e-02
  -5.96444927e-03   5.17033476e-03   3.31889841e-02  -3.24167082e-02]
```

LDA

LDA 的帮助几乎没有, 后文取消了基于 LDA 的特征

In [17]:

```
lda = LatentDirichletAllocation(n_components=10, max_iter=10, random_state=609)

lda_matrix = lda.fit_transform(tfidf_matrix)
```

```
/opt/conda/lib/python3.5/site-packages/sklearn/decomposition/online_lda.py:536: DeprecationWarning: The default valu
  DeprecationWarning)
```

目
录

Pretrained Word Vector

这里用到了预训练好的 Word Vector.

我尝试了 Glove.6B.100d, Glove.840B.300d, Word2vec.300d, Fasttext.300d。测试的结果是，维数越多越好；Glove 优于 Fasttext 优于 Word2Vec（应该此处的使用方式有关）

In [13]:

```
def create_lookup_table(filename):

    from collections import defaultdict

    with open(filename,'r') as file:
        lines = file.readlines()[1:]

    # 文本是以 Gensim 的 word2vec 格式保存的
    # 不从文本直接构建 Gensim word2vec 模型的原因是:
    # 预训练的词向量无法覆盖数据集中所有单词, 这里设置全为 0 的默认向量
    embedding = defaultdict(lambda: np.zeros((300,), dtype=np.float32))
    for line in lines:
        parts = line.split()
        # key is string word, value is numpy array for vector
        embedding[parts[0]] = np.asarray(parts[1:], dtype='float32')
    return embedding

lookup_table = create_lookup_table("/home/kesci/input/word2vec3861/simple.txt")
```

In [14]:

```
def sent2vec(sent):
    words = [w for w in text_to_word_sequence(sent) if w not in stopwords_eng]

    M = []
    for w in words:
        try:
            M.append(lookup_table[w])
        except:
            continue
    M = np.array(M)
    v = M.sum(axis=0)

    return v / np.sqrt((v ** 2).sum())
```

In [15]:

```
# 此处加载模型的目的仅仅是为了计算 word mover distance
keyedvectors = KeyedVectors.load_word2vec_format("/home/kesci/input/word2vec3861/simple.txt")

def wmd(s1, s2):
    s1 = text_to_word_sequence(s1)
    s2 = text_to_word_sequence(s2)
    s1 = [w for w in s1 if w not in stopwords_eng]
    s2 = [w for w in s2 if w not in stopwords_eng]
    return keyedvectors.wmdistance(s1, s2)
```

In [16]:

```
norm_keyedvectors = KeyedVectors.load_word2vec_format("/home/kesci/input/word2vec3861/simple.txt")
norm_keyedvectors.init_sims(replace=True)

def norm_wmd(s1, s2):
    s1 = text_to_word_sequence(s1)
    s2 = text_to_word_sequence(s2)
    s1 = [w for w in s1 if w not in stopwords_eng]
    s2 = [w for w in s2 if w not in stopwords_eng]
    return norm_keyedvectors.wmdistance(s1, s2)
```

In [27]:

```
train = pd.DataFrame()


train["len_s1"] = df_train.sent1.apply(lambda x: len(x))
train["len_s2"] = df_train.sent2.apply(lambda x: len(x))
train["diff_len"] = train.len_s1 - train.len_s2
train["len_char_s1"] = df_train.sent1.apply(lambda x: len("".join(set(x.replace(" ", "")))))
train["len_char_s2"] = df_train.sent2.apply(lambda x: len("".join(set(x.replace(" ", "")))))
train["len_word_s1"] = df_train.sent1.apply(lambda x: len(x.split()))
train["len_word_s2"] = df_train.sent2.apply(lambda x: len(x.split()))
train["common_words"] = df_train.apply(lambda x: len(set(x["sent1"].lower().split()).intersection(
set(x["sent2"].lower().split()))), axis=1)

# 帮助不大, 结果略微下降
# train["levenshtein"] = df_train.apply(lambda x: Levenshtein.distance(x.sent1, x.sent2), axis=1)
```

目
录

```
# # train["hamming"] = df_train.apply(lambda x: Levenshtein.hamming(x.sent1, x.sent2), axis=1)
# train["jaro"] = df_train.apply(lambda x: Levenshtein.jaro(x.sent1, x.sent2), axis=1)
# train["jaro_winkler"] = df_train.apply(lambda x: Levenshtein.jaro_winkler(x.sent1, x.sent2), axis=1)
# train["ratio"] = df_train.apply(lambda x: Levenshtein.ratio(x.sent1, x.sent2), axis=1)
# # train["seqratio"] = df_train.apply(lambda x: Levenshtein.ratio(word_sequences(x.sent1), word_sequences(x.sent2

ds_train = np.array([document_similarity(s1, s2) for s1, s2 in zip(df_train.sent1, df_train.sent2)], dtype="float3
train["path_similarity"] = ds_train[:, 0]
train["wup_similarity"] = ds_train[:, 1]


train["fuzz_qratio"] = df_train.apply(lambda x: fuzz.QRatio(x["sent1"], x["sent2"]), axis=1)
train["fuzz_wratio"] = df_train.apply(lambda x: fuzz.WRatio(x["sent1"], x["sent2"]), axis=1)
train["fuzz_partial_ratio"] = df_train.apply(lambda x: fuzz.partial_ratio(x["sent1"], x["sent2"]), axis=1)
train["fuzz_partial_token_set_ratio"] = df_train.apply(lambda x: fuzz.partial_token_set_ratio(x["sent1"], x["sent2
train["fuzz_partial_token_sort_ratio"] = df_train.apply(lambda x: fuzz.partial_token_sort_ratio(x["sent1"], x["sen
train["fuzz_token_set_ratio"] = df_train.apply(lambda x: fuzz.token_set_ratio(x["sent1"], x["sent2"]), axis=1)
train["fuzz_token_sort_ratio"] = df_train.apply(lambda x: fuzz.token_sort_ratio(x["sent1"], x["sent2"]), axis=1)

train["sv1"] = df_train.sent1.apply(sent2vec)
train["sv2"] = df_train.sent2.apply(sent2vec)


train["wmd"] = df_train.apply(lambda x: wmd(x.sent1, x.sent2), axis=1)
train["norm_wmd"] = df_train.apply(lambda x: norm_wmd(x.sent1, x.sent2), axis=1)
train["cosine"] = train.apply(lambda x: distance.cosine(x.sv1, x.sv2), axis=1)
train["manhattan"] = train.apply(lambda x: distance.cityblock(x.sv1, x.sv2), axis=1)
train["jaccard"] = train.apply(lambda x: distance.jaccard(x.sv1, x.sv2), axis=1)
train["canberra"] = train.apply(lambda x: distance.canberra(x.sv1, x.sv2), axis=1)
train["euclidean"] = train.apply(lambda x: distance.euclidean(x.sv1, x.sv2), axis=1)
train["minkowski"] = train.apply(lambda x: distance.minkowski(x.sv1, x.sv2), axis=1)
train["braycurtis"] = train.apply(lambda x: distance.braycurtis(x.sv1, x.sv2), axis=1)
train["doc2vec_sim"] = np.array([doc2vec.docvecs.similarity(i1, i2) for i1, i2 in zip(range(0, 1500), range(1500, 
train["cosine_tfidf"] = np.array([pairwise.cosine_distances(tfidf_matrix[i1], tfidf_matrix[i2])[0][0] for i1, i2 i
train["manhattan_tfidf"] = np.array([pairwise.manhattan_distances(tfidf_matrix[i1], tfidf_matrix[i2])[0][0] for i1
train["euclidean_tfidf"] = np.array([pairwise.euclidean_distances(tfidf_matrix[i1], tfidf_matrix[i2])[0][0] for i1
train["cosine_lsa"] = np.array([distance.cosine(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(0, 1500), 
train["manhattan_lsa"] = np.array([distance.cityblock(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(0, 1
train["jaccard_lsa"] = np.array([distance.jaccard(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(0, 1500)
train["canberra_lsa"] = np.array([distance.canberra(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(0, 150
train["euclidean_lsa"] = np.array([distance.euclidean(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(0, 1
train["minkowski_lsa"] = np.array([distance.minkowski(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(0, 1
train["braycurtis_lsa"] = np.array([distance.braycurtis(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(0,

# 基于 LDA 的特征反而是有害的
# train["cosine_lda"] = np.array([distance.cosine(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(0, 1500)
# train["manhattan_lda"] = np.array([distance.cityblock(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(0,
# train["jaccard_lda"] = np.array([distance.jaccard(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(0, 150
# train["canberra_lda"] = np.array([distance.canberra(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(0, 1
# train["euclidean_lda"] = np.array([distance.euclidean(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(0,
# train["minkowski_lda"] = np.array([distance.minkowski(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(0,
# train["braycurtis_lda"] = np.array([distance.braycurtis(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(

train["skew_s1"] = train.sv1.apply(stats.skew)
train["skew_s2"] = train.sv2.apply(stats.skew)
train["kurtosis_s1"] = train.sv1.apply(stats.kurtosis)
train["kurtosis_s2"] = train.sv2.apply(stats.kurtosis)
```

In [28]:

```
test = pd.DataFrame()

test["len_s1"] = df_test.sent1.apply(lambda x: len(x))
test["len_s2"] = df_test.sent2.apply(lambda x: len(x))
test["diff_len"] = test.len_s1 - test.len_s2
test["len_char_s1"] = df_test.sent1.apply(lambda x: len("".join(set(x.replace(" ", "")))))
test["len_char_s2"] = df_test.sent2.apply(lambda x: len("".join(set(x.replace(" ", "")))))
test["len_word_s1"] = df_test.sent1.apply(lambda x: len(x.split()))
test["len_word_s2"] = df_test.sent2.apply(lambda x: len(x.split()))
test["common_words"] = df_test.apply(lambda x: len(set(x["sent1"].lower().split()).intersection(
set(x["sent2"].lower().split()))), axis=1)


# 帮助不大，结果略微下降
# test["levenshtein"] = df_test.apply(lambda x: Levenshtein.distance(x.sent1, x.sent2), axis=1)
# # test["hamming"] = df_test.apply(lambda x: Levenshtein.hamming(x.sent1, x.sent2), axis=1)
# test["jaro"] = df_test.apply(lambda x: Levenshtein.jaro(x.sent1, x.sent2), axis=1)
# test["jaro_winkler"] = df_test.apply(lambda x: Levenshtein.jaro_winkler(x.sent1, x.sent2), axis=1)
# test["ratio"] = df_test.apply(lambda x: Levenshtein.ratio(x.sent1, x.sent2), axis=1)
# # test["seqratio"] = df_test.apply(lambda x: Levenshtein.ratio(word_sequences(x.sent1), word_sequences(x.sent2))

ds_test = np.array([document_similarity(s1, s2) for s1, s2 in zip(df_test.sent1, df_test.sent2)], dtype="float32")
test["path_similarity"] = ds_test[:, 0]
```

```
test["wup_similarity"] = ds_test[:, 1]


test["fuzz_qratio"] = df_test.apply(lambda x: fuzz.QRatio(x["sent1"], x["sent2"]), axis=1)
test["fuzz_wratio"] = df_test.apply(lambda x: fuzz.WRatio(x["sent1"], x["sent2"]), axis=1)
test["fuzz_partial_ratio"] = df_test.apply(lambda x: fuzz.partial_ratio(x["sent1"], x["sent2"]), axis=1)
test["fuzz_partial_token_set_ratio"] = df_test.apply(lambda x: fuzz.partial_token_set_ratio(x["sent1"], x["sent2"]
test["fuzz_partial_token_sort_ratio"] = df_test.apply(lambda x: fuzz.partial_token_sort_ratio(x["sent1"], x["sent2
test["fuzz_token_set_ratio"] = df_test.apply(lambda x: fuzz.token_set_ratio(x["sent1"], x["sent2"]), axis=1)
test["fuzz_token_sort_ratio"] = df_test.apply(lambda x: fuzz.token_sort_ratio(x["sent1"], x["sent2"]), axis=1)

test["sv1"] = df_test.sent1.apply(sent2vec)
test["sv2"] = df_test.sent2.apply(sent2vec)

test["wmd"] = df_test.apply(lambda x: wmd(x.sent1, x.sent2), axis=1)
test["norm_wmd"] = df_test.apply(lambda x: norm_wmd(x.sent1, x.sent2), axis=1)
test["cosine"] = test.apply(lambda x: distance.cosine(x.sv1, x.sv2), axis=1)
test["manhattan"] = test.apply(lambda x: distance.cityblock(x.sv1, x.sv2), axis=1)
test["jaccard"] = test.apply(lambda x: distance.jaccard(x.sv1, x.sv2), axis=1)
test["canberra"] = test.apply(lambda x: distance.canberra(x.sv1, x.sv2), axis=1)
test["euclidean"] = test.apply(lambda x: distance.euclidean(x.sv1, x.sv2), axis=1)
test["minkowski"] = test.apply(lambda x: distance.minkowski(x.sv1, x.sv2), axis=1)
test["braycurtis"] = test.apply(lambda x: distance.braycurtis(x.sv1, x.sv2), axis=1)
test["doc2vec_sim"] = np.array([doc2vec.docvecs.similarity(i1, i2) for i1, i2 in zip(range(3000, 3750), range(3750
test["cosine_tfidf"] = np.array([pairwise.cosine_distances(tfidf_matrix[i1], tfidf_matrix[i2])[0][0] for i1, i2 in
test["manhattan_tfidf"] = np.array([pairwise.manhattan_distances(tfidf_matrix[i1], tfidf_matrix[i2])[0][0] for i1,
test["euclidean_tfidf"] = np.array([pairwise.euclidean_distances(tfidf_matrix[i1], tfidf_matrix[i2])[0][0] for i1,
test["cosine_lsa"] = np.array([distance.cosine(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(3000, 3750)
test["manhattan_lsa"] = np.array([distance.cityblock(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(3000,
test["jaccard_lsa"] = np.array([distance.jaccard(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(3000, 375
test["canberra_lsa"] = np.array([distance.canberra(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(3000, 3
test["euclidean_lsa"] = np.array([distance.euclidean(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(3000,
test["minkowski_lsa"] = np.array([distance.minkowski(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(3000,
test["braycurtis_lsa"] = np.array([distance.braycurtis(lsa_matrix[i1], lsa_matrix[i2]) for i1, i2 in zip(range(300

# 基于 LDA 的特征反而是有害的
# test["cosine_lda"] = np.array([distance.cosine(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(3000, 375
# test["manhattan_lda"] = np.array([distance.cityblock(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(300
# test["jaccard_lda"] = np.array([distance.jaccard(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(3000, 3
# test["canberra_lda"] = np.array([distance.canberra(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(3000,
# test["euclidean_lda"] = np.array([distance.euclidean(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(300
# test["minkowski_lda"] = np.array([distance.minkowski(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(300
# test["braycurtis_lda"] = np.array([distance.braycurtis(lda_matrix[i1], lda_matrix[i2]) for i1, i2 in zip(range(3


test["skew_s1"] = test.sv1.apply(stats.skew)
test["skew_s2"] = test.sv2.apply(stats.skew)
test["kurtosis_s1"] = test.sv1.apply(stats.kurtosis)
test["kurtosis_s2"] = test.sv2.apply(stats.kurtosis)
```

In [33]:

```
train_no_vec = train.drop(["sv1", "sv2"], axis=1)  # 不使用向量作为输入，因此去掉
test_no_vec = test.drop(["sv1", "sv2"], axis=1)
```

In [34]:

```
std = StandardScaler()

train_std = std.fit_transform(train_no_vec)
test_std = std.transform(test_no_vec)

x_train, x_valid, y_train, y_valid = train_test_split(train_std, df_train["similarity"], test_size=0.1, random_sta
```

In [55]:

```
xgbr_params = {
    "objective": "reg:linear",
    "learning_rate": 0.06,  # 0.01 ~ 0.2
    "max_depth": 4,  # 3 ~ 10; 4 may be the best
    "seed": 609,
    "n_estimators": 500
}

xgbr_train = xgb.DMatrix(x_train, label=y_train)
xgbr_valid = xgb.DMatrix(x_valid, label=y_valid)

watchlist = [(xgbr_train, "train"), (xgbr_valid, "valid")]

xgbr = xgb.XGBRegressor(**xgbr_params)
```

In [56]:

```
xgbr.fit(x_train, y_train, early_stopping_rounds=50, eval_metric="rmse", eval_set=[(x_valid, y_valid)], verbose=10
xgbr_pred = xgbr.predict(test_std)
```

```
xgbr_pred = xgbr.predict(test_std)

print("There are some prediction results less than 0:", np.any(xgbr_pred < 0))
print("There are some prediction results greater than 5:", np.any(xgbr_pred > 5))
print("Outliers:", [i for i in xgbr_pred if i < 0 or i > 5])
```

目
录

```
[0]     validation_0-rmse:2.23125
Will train until validation_0-rmse hasn't improved in 50 rounds.
[10]    validation_0-rmse:1.41145
[20]    validation_0-rmse:1.0545
[30]    validation_0-rmse:0.909618
[40]    validation_0-rmse:0.848068
[50]    validation_0-rmse:0.821394
[60]    validation_0-rmse:0.806166
[70]    validation_0-rmse:0.799396
[80]    validation_0-rmse:0.795957
[90]    validation_0-rmse:0.79582
[100]   validation_0-rmse:0.79528
[110]   validation_0-rmse:0.796483
[120]   validation_0-rmse:0.798296
[130]   validation_0-rmse:0.795915
[140]   validation_0-rmse:0.798184
[150]   validation_0-rmse:0.798101
Stopping. Best iteration:
[100]   validation_0-rmse:0.79528

There are some prediction results less than 0: True
There are some prediction results greater than 5: True
Outliers: [5.0332413, -0.022093177, -0.031235278, -0.10757601]
```

In [57]:

```
svr = SVR("rbf", C=1.0, gamma=0.01)
```

In [58]:

```
svr.fit(train_std, df_train["similarity"])
svr_pred = svr.predict(test_std)
print("There are some prediction results less than 0:", np.any(svr_pred < 0))
print("There are some prediction results greater than 5:", np.any(svr_pred > 5))
print("Outliers", [i for i in svr_pred if i < 0 or i > 5])
```

```
There are some prediction results less than 0: True
There are some prediction results greater than 5: False
Outliers [-0.03703363135005322, -0.078863672062461365, -0.11666086199010905, -0.045072042974326987, -0.030072957381
```

In [59]:

```
rfr = RandomForestRegressor(max_depth=10, n_estimators=35)
```

In [60]:

```
rfr.fit(train_std, df_train["similarity"])
rfr_pred = rfr.predict(test_std)
print("There are some prediction results less than 0:", np.any(rfr_pred < 0))
print("There are some prediction results greater than 5:", np.any(rfr_pred > 5))
print("Outliers", [i for i in rfr_pred if i < 0 or i > 5])
```

```
There are some prediction results less than 0: False
There are some prediction results greater than 5: False
Outliers []
```

In [61]:

```
gbr = GradientBoostingRegressor(learning_rate=0.1, max_depth=4, n_estimators=50)
```

In [62]:

```
gbr.fit(train_std, df_train["similarity"])
gbr_pred = gbr.predict(test_std)
print("There are some prediction results less than 0:", np.any(gbr_pred < 0))
print("There are some prediction results greater than 5:", np.any(gbr_pred > 5))
print("Outliers", [i for i in gbr_pred if i < 0 or i > 5])
```

```
There are some prediction results less than 0: True
There are some prediction results greater than 5: False
Outliers [-0.011839752960001195]
```

In [63]:

```
%reload_ext lightgbm

lgbr_params = {
    "objective": "regression",
    "learning_rate": 0.05,
    "num_iteration": 1000,
```

目
录

```
        "boosting_type": "gbdt",
        "num_leaves": 5,
        "max_bin": 30,
        "bagging_fraction": 1,
        "feature_fraction": 0.7,
        "bagging_seed": 9,
        "feature_fraction_seed": 9,
        "min_data_in_leaf": 16,
        "metric": "root_mean_squared_error",
    }

    lgbr = lgb.LGBMRegressor(**lgbr_params)
```

In [64]:

```
lgbr.fit(x_train, y_train, early_stopping_rounds=100, eval_set=[(x_valid, y_valid)], verbose=10)
lgbr_pred = lgbr.predict(test_std)
print("There are some prediction results less than 0:", np.any(lgbr_pred < 0))
print("There are some prediction results greater than 5:", np.any(lgbr_pred > 5))
print("Outliers:", [i for i in lgbr_pred if i < 0 or i > 5])
```

```
/opt/conda/lib/python3.5/site-packages/lightgbm/engine.py:99: UserWarning: Found `num_iteration` in params. Will use
  warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))
/opt/conda/lib/python3.5/site-packages/lightgbm/basic.py:642: UserWarning: max_bin keyword has been found in `params
  'Please use {0} argument of the Dataset constructor to pass this parameter.'.format(key))
/opt/conda/lib/python3.5/site-packages/lightgbm/basic.py:648: LGBMDeprecationWarning: The `max_bin` parameter is dep
  'Please use `params` to pass this parameter.', LGBMDeprecationWarning)

Training until validation scores don't improve for 100 rounds.
[10]	valid_0's l2: 1.57614
[20]	valid_0's l2: 1.12403
[30]	valid_0's l2: 0.928739
[40]	valid_0's l2: 0.836315
[50]	valid_0's l2: 0.794365
[60]	valid_0's l2: 0.7603
[70]	valid_0's l2: 0.742814
[80]	valid_0's l2: 0.730852
[90]	valid_0's l2: 0.724512
[100]	valid_0's l2: 0.712786
[110]	valid_0's l2: 0.709167
[120]	valid_0's l2: 0.703302
[130]	valid_0's l2: 0.700201
[140]	valid_0's l2: 0.693613
[150]	valid_0's l2: 0.690859
[160]	valid_0's l2: 0.688017
[170]	valid_0's l2: 0.687172
[180]	valid_0's l2: 0.684123
[190]	valid_0's l2: 0.682293
[200]	valid_0's l2: 0.683489
[210]	valid_0's l2: 0.681991
[220]	valid_0's l2: 0.68339
[230]	valid_0's l2: 0.681289
[240]	valid_0's l2: 0.679198
[250]	valid_0's l2: 0.680684
[260]	valid_0's l2: 0.675905
[270]	valid_0's l2: 0.678571
[280]	valid_0's l2: 0.678299
[290]	valid_0's l2: 0.68008
[300]	valid_0's l2: 0.681442
[310]	valid_0's l2: 0.679436
[320]	valid_0's l2: 0.67821
[330]	valid_0's l2: 0.678727
[340]	valid_0's l2: 0.677536
[350]	valid_0's l2: 0.677293
[360]	valid_0's l2: 0.675022
[370]	valid_0's l2: 0.674782
[380]	valid_0's l2: 0.671973
[390]	valid_0's l2: 0.670401
[400]	valid_0's l2: 0.671418
[410]	valid_0's l2: 0.671664
[420]	valid_0's l2: 0.669364
[430]	valid_0's l2: 0.670659
[440]	valid_0's l2: 0.671367
[450]	valid_0's l2: 0.672853
[460]	valid_0's l2: 0.6735
[470]	valid_0's l2: 0.672558
[480]	valid_0's l2: 0.673004
[490]	valid_0's l2: 0.67576
[500]	valid_0's l2: 0.675577
[510]	valid_0's l2: 0.677928
Early stopping, best iteration is:
[418]	valid_0's l2: 0.669298
There are some prediction results less than 0: True
```

```
There are some prediction results greater than 5: True
Outliers: [-0.0057286077361911741, -0.02294872671885885, -0.01293983648134801, 5.1113544046713608, -0.15301329001598
```

我使用了 2 种计算均值的方法:

1. 对预测值做截断, 即直接将预测值超出 0~5 范围的异常值设置为 0 或 5, 再求平均;
2. 对每条数据对应的预测值, 去掉最大值和最小值, 求平均, 然后根据是否超出边界再截断.

> 提交结果表明, 方法二并没有使结果更好

In [65]:

```python
# 方法一（懒人写法）

with open("submission_sample", "w") as f:
    for idx, p, s, r, g, l in zip(df_test["id"], xgbr_pred, svr_pred, rfr_pred, gbr_pred, lgbr_pred):
        if p > 5:
            p = 5.0
        elif p < 0:
            p = 0
        if s > 5:
            s = 5.0
        elif s < 0:
            s = 0
        if r > 5:
            r = 5.0
        elif r < 0:
            r = 0
        if g > 5:
            g = 5.0
        elif g < 0:
            g = 0
        if l > 5:
            l = 5.0
        elif l < 0:
            l = 0
        f.write(str(idx) + "," + str(sum([p, s,r, g, l]) / 5.0) + "\n")
```

In [75]:

```python
# 方法二

def smooth_average(nums):
    """平滑均值，去掉最大值和最小值，再求平均.（结果反而更差了.）"""
    return (sum(nums) - max(nums) - min(nums)) / (len(nums) - 2)

with open("submission_sample", "w") as f:
    for idx, x, s, r, g, l, p in zip(df_test["id"], xgbr_pred, svr_pred, rfr_pred, gbr_pred, lgbr_pred, stacker_pr
        average = smooth_average([x, s, r, g, l, p])
        if average > 5.0:
            average = 5.0
        elif average < 0.0:
            average = 0.0
        f.write(str(idx) + "," + str(average) + "\n")
```

In [76]:

```
!head submission_sample -n 5
```

```
11501,2.3646556983
11502,3.03644582524
11503,1.13898102434
11504,1.16023804212
11505,2.49692323963
```

In [77]:

```
!wget -nv -O kesci_submit https://cdn.kesci.com/submit_tool/v1/kesci_submit&&chmod +x kesci_submit
```

```
2018-02-02 15:29:52 URL:https://cdn.kesci.com/submit_tool/v1/kesci_submit [7840472/7840472] -> "kesci_submit" [1]
```

In [78]:

```
!./kesci_submit -token balbalbalba -file submission_sample
```

```
Kesci Submit Tool
Working...
Success.
OK
```

本项目的一个启示是： 特征有多重要，几乎是特征堆出来的 0.85！

总共三四十个特征，此时再用深度学习的话，结果应该会好很多，有兴趣的同学欢迎尝试。

目
录

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/DpNWI/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/)（本文一开始的一些特征是来自此文，很强）
- 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)（不会传统方法的我，都是看文档，对着API写的）
- 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)

附录

A. GridSearch 调参

以下是 `GridSearch` 搜索 SVR, RandomForestRegressor, GradientBoostingRegressor 的较优参数的代码.

搜索时间与参数的取值范围以及参数个数成比例, 增大参数取值范围或参数个数, 搜索时间将成倍增加唉, 因此只选择了少量参数, 并粗糙地设置了参数值。我在本地的搜索已经缩小了参数范围。

In [ ]:

```
pipe_svr = Pipeline([
    ("svr", SVR())
])
# 初步确定比较优的参数范围是: (kernel 仅选择了 linear 与 rbf; 初始范围 [0.0001, 10])
# 'svr__gamma': 0.01, 'svr__kernel': 'rbf', 'svr__C': 1.0
param_range = [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0]
param_grid = [
    {
        "svr__C": param_range,
        "svr__gamma": param_range,
        "svr__kernel": ["rbf"]
    }
]
gs = GridSearchCV(estimator=pipe_svr,
                  param_grid=param_grid,
                  # scoring="mse",
                  cv=10,
                  verbose=2,
                  n_jobs=-1)
gs = gs.fit(train_std, df_train["similarity"])

print(gs.best_score_)
print(gs.best_params_)
svr = gs.best_estimator_
```

In [ ]:

```
pipe_rfr = Pipeline([
    ("rfr", RandomForestRegressor(random_state=609))
])
num_estimator_range = [30, 35, 40, 45, 50]
depth_range = [8, 9, 10, 11, 12, 13, 14]
param_grid = [
    {
    }
]
```

目
录

```
gs = GridSearchCV(estimator=pipe_rfr,
                  param_grid=param_grid,
                  cv=10,
                  verbose=2,
                  n_jobs=-1)
gs = gs.fit(train_std, df_train["similarity"])

print(gs.best_score_)
print(gs.best_params_)
rfr = gs.best_estimator_
```

In [ ]:

```
pipe_gbr = Pipeline([
    ("gbr", GradientBoostingRegressor(random_state=609))
])
num_estimator_range = [40, 50, 60]
lr_range = [0.05, 0.1, 0.2]
depth_range = [3, 4, 5]

param_grid = [
    {
        "gbr__max_depth": depth_range,
        "gbr__n_estimators": num_estimator_range,
        "gbr__learning_rate": lr_range,
    }
]
gs = GridSearchCV(estimator=pipe_gbr,
                  param_grid=param_grid,
                  cv=10,
                  verbose=2,
                  n_jobs=-1)
gs = gs.fit(train_std, df_train["similarity"])

print(gs.best_score_)
print(gs.best_params_)
gbr = gs.best_estimator_
```

B. Stacking

参考助教-常永炷的 Stacking 方法, 但预测结果反而不如求所有 Regressors 预测值的均值.

此时的预测值和其他 Regressors 的值偏差较大, 即使再求均值, 影响也是负面的.

应该是我参数没调好.

In [69]:

```
regressors = [rfr, svr, gbr, lgbr,xgbr]
stacker = StackingRegressor(regressors=regressors, meta_regressor=xgbr, verbose=1)
```

In [70]:

```
stacker.fit(x_train, y_train)
stacker_pred = stacker.predict(test_std)

print("There are some prediction results less than 0:", np.any(stacker_pred < 0))
print("There are some prediction results greater than 5:", np.any(stacker_pred > 5))
print("Outliers:", [i for i in stacker_pred if i < 0 or i > 5])
```

```
Fitting 5 regressors...
Fitting regressor1: randomforestregressor (1/5)
Fitting regressor2: svr (2/5)
Fitting regressor3: gradientboostingregressor (3/5)
Fitting regressor4: lgbmregressor (4/5)

/opt/conda/lib/python3.5/site-packages/lightgbm/engine.py:99: UserWarning: Found `num_iteration` in params. Will use
  warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))
/opt/conda/lib/python3.5/site-packages/lightgbm/basic.py:642: UserWarning: max_bin keyword has been found in `params
  'Please use {0} argument of the Dataset constructor to pass this parameter.'.format(key))
/opt/conda/lib/python3.5/site-packages/lightgbm/basic.py:648: LGBMDeprecationWarning: The `max_bin` parameter is dep
  'Please use `params` to pass this parameter.', LGBMDeprecationWarning)

Fitting regressor5: xgbregressor (5/5)
There are some prediction results less than 0: True
There are some prediction results greater than 5: True
Outliers: [-0.0005095005, 5.0011554, 5.0019403, 5.0034137, -0.00013130903, -0.00062811375, -0.00088179111, 5.021609
```

In [71]:

```
with open("submission_sample", "w") as f:
    for idx, p in zip(df_test["id"], stacker_pred):
        if p > 5:
            p = 5.0
```

```
        elif p < 0:
            p = 0
        f.write(str(idx) + "," + str(p) + "\n")
```

目

录

In [ ]:

目

录

目

录

和鲸社区公众号

工具 内容 实战

K-Lab 项目 比赛

数据集 众包

专栏

专区

商务合作