时效性项目中训练数据分析方法

关键词

- 文本分类、主动学习、训练数据分析

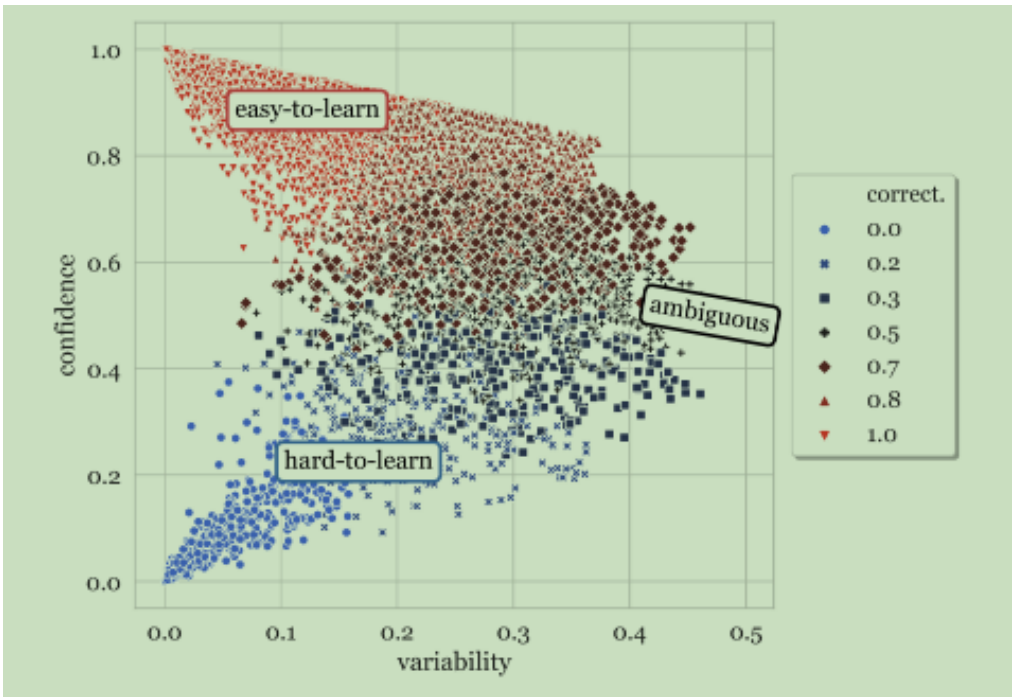
一、概述

- 在知乎的推荐类分发场景中，内容的时效性是一个非常重要的属性，我们应该在合适的时间，给合适的人，推荐合适的内容。所谓合适的内容属性之一，就是有合适的时效性。比如此时此刻（2022 年 12 月 08 日）推荐（如何看待今年（2021）618 大促活动），就会感觉非常奇怪，因为此内容已经过时效了
- 根据上述描述，我们需要给内容打上一个时效性标签。根据我们之前的调研，一般情况下，主要采用的是文本分类的技术
- 就文本分类技术而言，是一个比较成熟的技术。作为算法工程师，我们提升效果的方法，一般会做更大更深的模型，或者使用大量的训练数据。但受限于有限的资源，我们期望有一种更优更快的方法，能迅速的提升效果
- 在时效性项目中
    - 针对模型方面，我们考虑到领域特征是时效性标签非常重要的影响因素，因此使用了多（同构）任务学习的方法，给模型效果带来了一定的提升
    - 针对数据方面，我们使用了训练数据分析的方法，去找到当前训练样本中的容易学习的样本、容易混淆的样本和难以学习的样本。通过对训练样本进行分类，我们可以清晰地看到每一类样本在训练过程和训练结果中的作用，可以根据自己的需求，针对性的补充某类数据，来达到预定的目标
- 我们下面的讨论，主要集中在训练数据分析方法中，不会对上述的多（同构）任务学习作太多的讨论

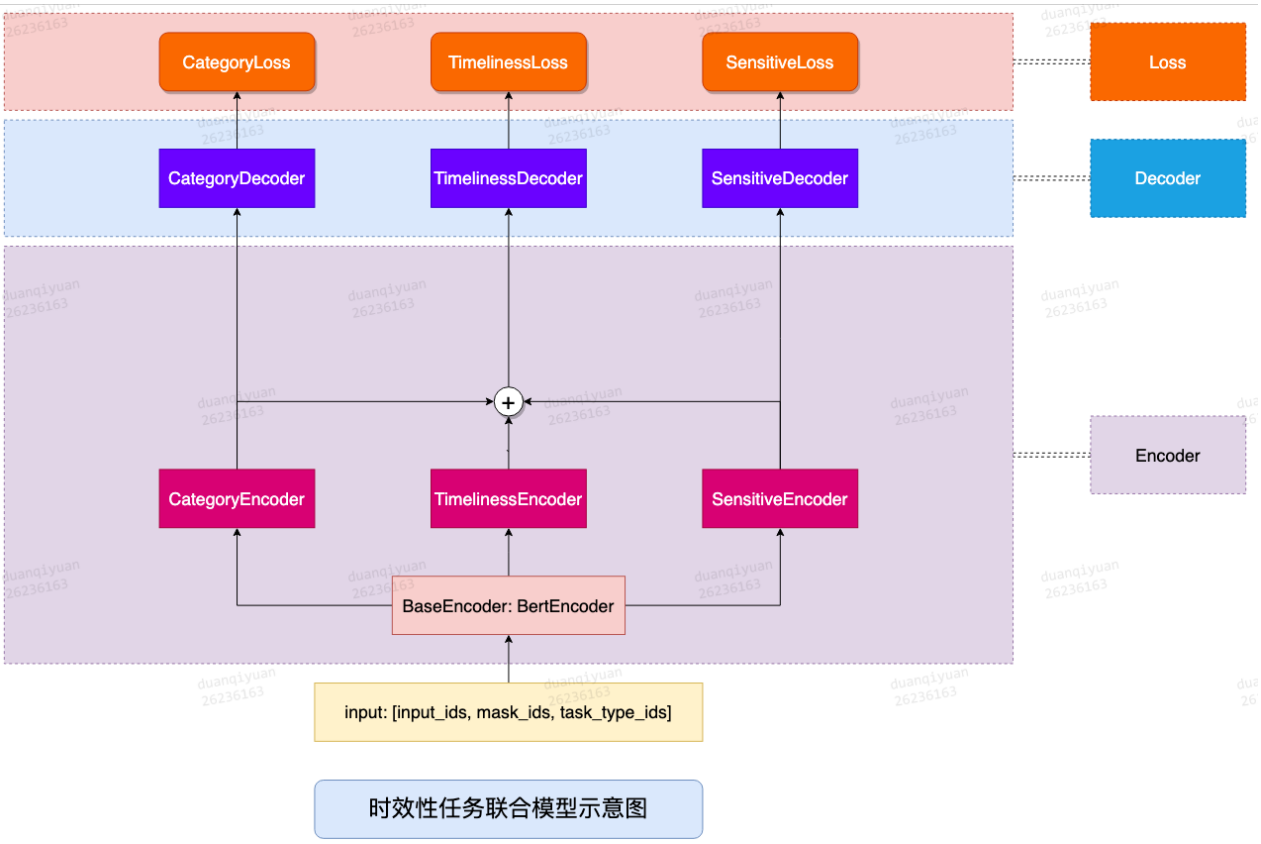
二、方法

- 主动学习
    - 假设
        - 数据层
            - 抽象假设
                - 不同的数据样本对于某项具体任务的作用和价值是不一样的
            - 具象假设
                - 数据密度说
                    - 用少量的样本，把握大量数据上的分布
        - 模型层
            - 抽象假设
                - 模型或者算法可以利用已经学到的一些知识，选择出对提升该模型效果有利的样本，从而减少训练数据量
            - 具象假设
                - 深度学习模型由于其复杂的结构，具有很强的学习容量

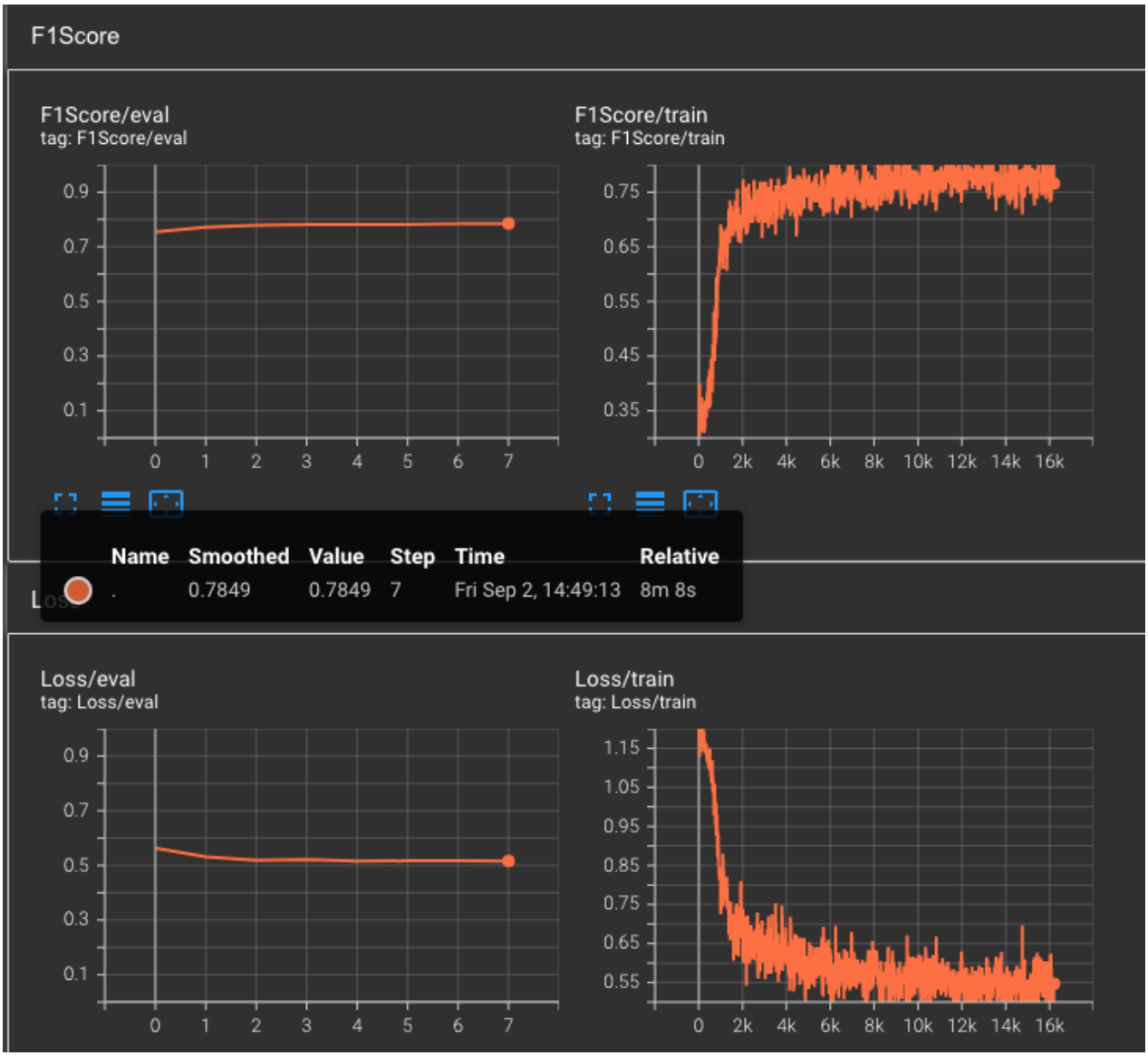
- 定义
  - 通过机器学习的方法获取到那些比较**“难”**分类的样本数据，让人工再次确认和审核，然后将人工标注得到的数据再次使用有监督学习模型或者半监督学习模型进行训练，逐步提升模型的效果，将人工经验融入机器学习的模型中
    - 这里所谓的**难**，需要根据不同的情况去定义
- 目标
  - 主动学习都是为了降低标注成本，迅速提升模型效果而存在的

- 训练数据的动态分析
  - 描述
    - 在模型训练过程中，模型会给每个样本在正确标签的一个置信度分数，随着 epoch 的增加，我们会针对同一个样本，或者多个置信度分数，我们可以根据这个置信度分数及其分布，来评估当前模型对某个样本的学习程度
  - 关键指标
    - 置信度分布中均值和方差
      - 均值体现了模型对某样本的整体把握
      - 方差表示二阶置信度，提现了模型对置信度的把握程度
  - 分类说明
    - 均值越大，方差越小，说明样本越容易学习，且模型很有把握
      - 这部分样本可以加快模型收敛，参考 Curriculum learning 思想
    - 均值越小，方法越小，说明样本不好学习，或者说模型对这部分样本有把握出现了错误样本
      - 这部分样本提升模型的泛化能力，可以用于清洗数据
    - 介于中间的样本，一般来说模型对这部分样本容易混淆
      - 这部分样本提升模型的泛化能力
  - 

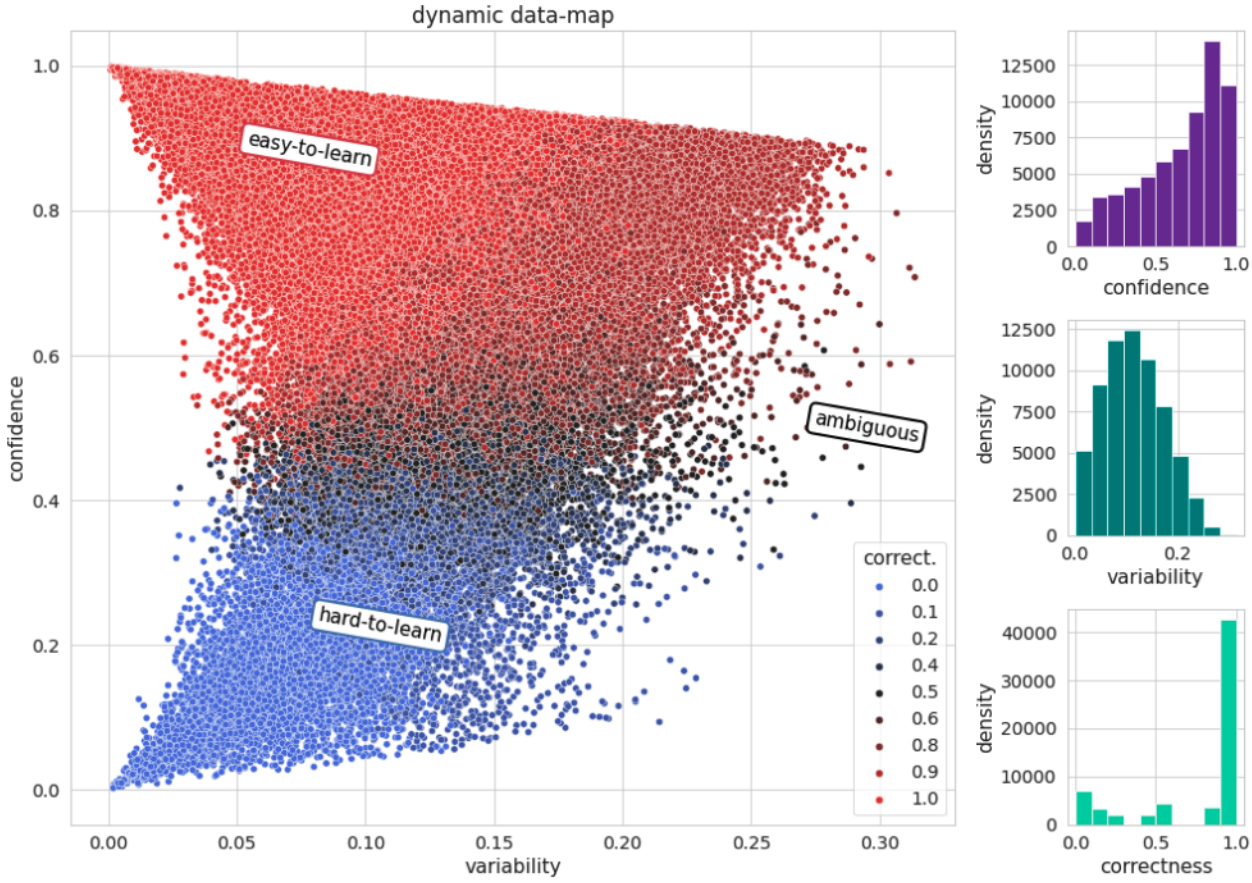
三、实验

- 时效性分类模型
  - Timeliness Loss 是主任务
  - 

    时效性任务联合模型示意图
- Baseline 效果

- 
F1Score
F1Score/eval
tag: F1Score/eval
F1Score/train
tag: F1Score/train
Name Smoothed Value Step Time Relative
. 0.7849 0.7849 7 Fri Sep 2, 14:49:13 8m 8s
Loss/eval
tag: Loss/eval
Loss/train
tag: Loss/train

- 训练样本数据分析
  -


dynamic data-map
easy-to-learn
ambiguous
hard-to-learn
confidence
variability
correct.
0.0
0.1
0.2
0.4
0.5
0.6
0.8
0.9
1.0
density
confidence
variability
correctness

-

```
cartographer.dynamics.head(20)
```

	sid	correctness	confidence	variability	forgetfulness	threshold_closeness
0	27092315	6	0.582323	0.174863	1	0.243223
1	66262582	7	0.879828	0.259515	0	0.105730
2	50662086	2	0.351870	0.094098	2	0.228058
3	59036977	7	0.727850	0.226737	0	0.198084
4	77012537	7	0.718523	0.198229	0	0.202248
5	47308149	7	0.747274	0.200250	0	0.188856
6	59573563	8	0.737514	0.169420	0	0.193587
7	1584623	8	0.915591	0.122405	0	0.077284
8	3447162	7	0.704334	0.200274	1	0.208248
9	84026785	1	0.071692	0.140048	1	0.066552
10	8381570	8	0.919353	0.183594	0	0.074143
11	31800059	7	0.873035	0.212212	0	0.110845
12	26852205	7	0.888678	0.225115	0	0.098930
13	33366794	5	0.505443	0.124373	2	0.249970
14	25151590	7	0.573018	0.111720	1	0.244668
15	41600142	7	0.512990	0.121540	0	0.249831
16	46270305	0	0.085609	0.064391	10000	0.078280
17	78632154	1	0.308333	0.187331	1	0.213264
18	47483973	8	0.811728	0.168377	0	0.152826
19	62847726	7	0.725393	0.197245	0	0.199198

- 详细分析
  - **问题定义**
    - 有哪些样本分析角度，分析角度的重要性是怎样的，每种角度体现怎样的特点，不同角度如何联系
    - 如何利用上述分析，辅助选择样本
      - 样本重标
      - 样本扩充
  - **要素拆解和概念结构**
    - Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics
    - Identifying Mislabeled Data using the Area Under the Margin Ranking
    - confidence
      - 模型对预测结果的置信度

    - variability
        - 模型对预测结果的置信度的置信度（二阶置信度），相对整个样本集上的平均置信度的偏移，这个值越小越好
    - correctness
        - 模型在多轮学习中，对某一样本预测的正确次数的统计，可以算一个整体正确率，体现的是模型对该样本的学习的能力，或者某样本是否是一个好学的样本，体现样本的难易程度
    - forgetfulness
        - 模型在多轮学习中，正确预测样本之后，又忘记所学的次数统计，体现的是样本的二阶难易度
    - threshold_cloness
        - confidence * (1 - confidence)，计算了一个置信度熵，越大越不好，说明置信度没啥意义
- **作出假设**
    - 假设的基本方向
        - 样本重标
            - 标注错误的样本
            - confidence 低、variability 低，说明很有把握这部分不是现有的标签
        - 样本扩充
            - 困难样本
            - confidence 中、variability 中，说明模型对这部分数据，不确定，且对预测结果的置信度也不确定
            - correctness 低，难样本
            - forgetfulness 高，更难的样本
            - correctness - forgetfulness 低，更难的样本
    - 具体假设
        - 错误样本剔除后提升效果
        - 错误样本修正后提升效果，且效果好于剔除
        - 难样本和更难样本扩充后，提升效果
        - 更难样本的扩充，效果提升更明显
- **研究设计**
    - 数据层
        - 重标样本
        - 扩充样本
    - 统计和观测层
        - 重标样本的分布

```
relabel_confidence_threshold = 0.3
relabel_confidence_variability = 0.15

relabel_df = cartographer.dynamics[
    (cartographer.dynamics['confidence'] <
relabel_confidence_threshold) &
```

```
    (cartographer.dynamics['variability'] <
relabel_confidence_variability) &
    (cartographer.dynamics['forgetfulness'] == 10000)
]

relabel_df_detail.value_counts("timeliness_label")
timeliness_label
1    2945
2    2018
0    1540
dtype: int64

发现【如何看待、如何评价、怎么看待、怎么评价】的数据，标签打的比较混乱，可以都设置为 1 标签
re_select =
relabel_df_detail['question_title'].str.contains("如何评价|如何看待|怎么看待|怎么评价")
re_select_df = relabel_df_detail[re_select]
re_select_df.value_counts("timeliness_label")
timeliness_label
1    1275
0     340
2      12
dtype: int64

剩余根据条件筛选出来的数据，发现 1 标签的没啥问题，只需要复标 0 和 2 标签的数据

最终剩余数据 61594
最终复标数据 3206
```

- 难样本的分布

```
diffcult_df = cartographer.dynamics[
    (0.35 < cartographer.dynamics['confidence']) &
    (cartographer.dynamics['confidence'] < 0.5) &
    (0.05 < cartographer.dynamics['variability']) &
    (cartographer.dynamics['variability'] < 0.18)
]

diffcult_df_detail.value_counts('timeliness_label')
timeliness_label
1    3420
2    1942
0    1095
dtype: int64

困难样本种子池 diffcult_df_detail 6457
```

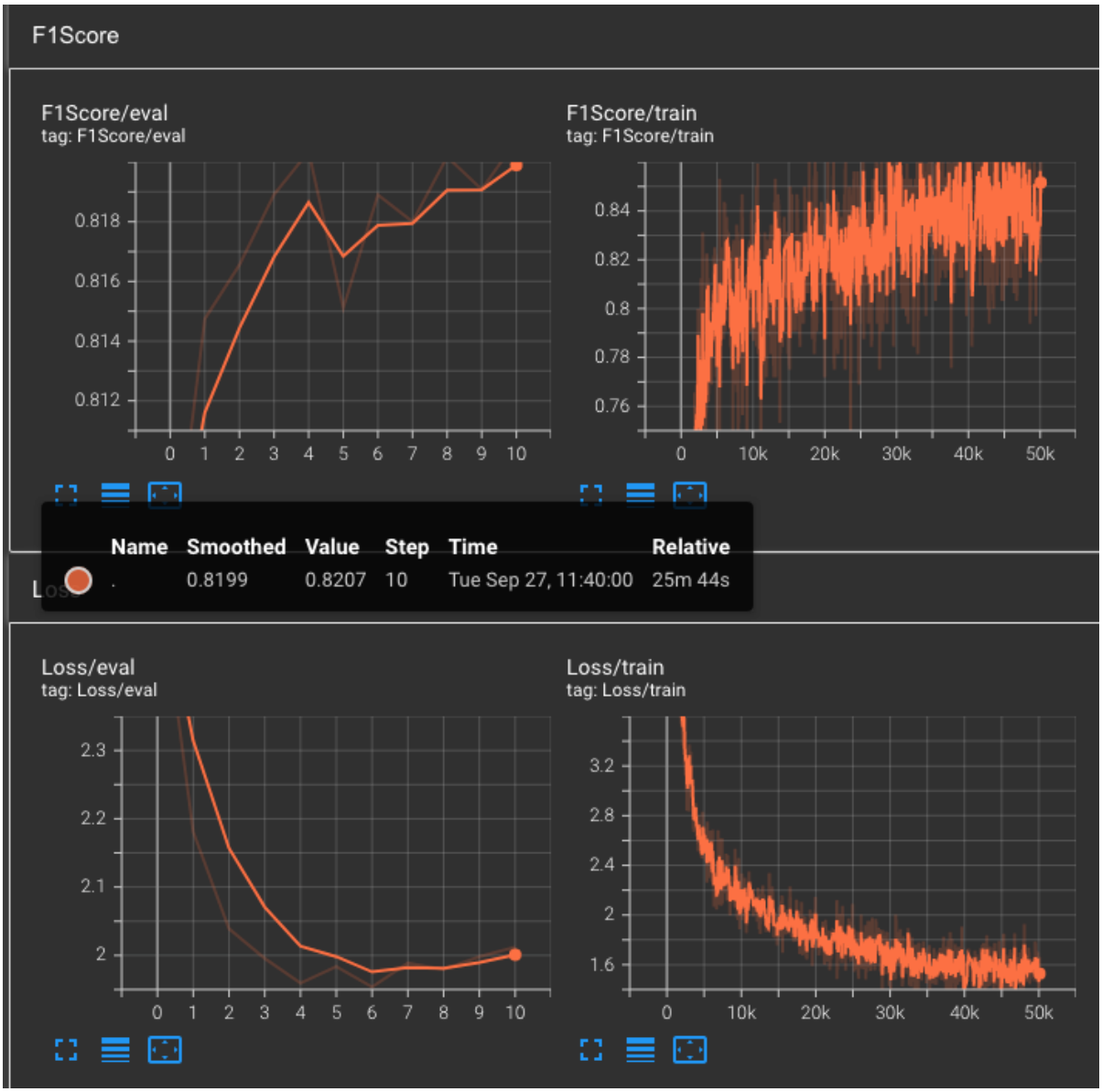
        - 更难样本的分布

```
very_diffcult_df_source = cartographer.dynamics[
    (cartographer.dynamics['correctness']) < 3 &
    (cartographer.dynamics['correctness'] -
cartographer.dynamics['forgetfulness'] < 2)
]

very_diffcult_df_detail = very_diffcult_df_source_detail[

(~very_diffcult_df_source_detail['question_id'].isin(diffcult_
df_detail['question_id'])) &

(~very_diffcult_df_source_detail['question_id'].isin(final_rel
abel_df['question_id'])) &

(~very_diffcult_df_source_detail['question_id'].isin(relabel_r
e_select_df['question_id']))
]

very_diffcult_df_detail.value_counts('timeliness_label')
timeliness_label
1    1864
2     137
0      39
dtype: int64

更困难样本种子池 very_diffcult_df_detail 2040
```

    - 操作层
        - 样本重标
            - 剔除后训练
            - 修正后训练
        - 样本扩充
            - 验证一下哪部分数据，收益更大
- 数据消融实验
    - **实验一【一般困难样本扩充】**

```
原始训练数据 104889 条
扩充困难数据 28788 条
合计数据 133677 条
```

        -

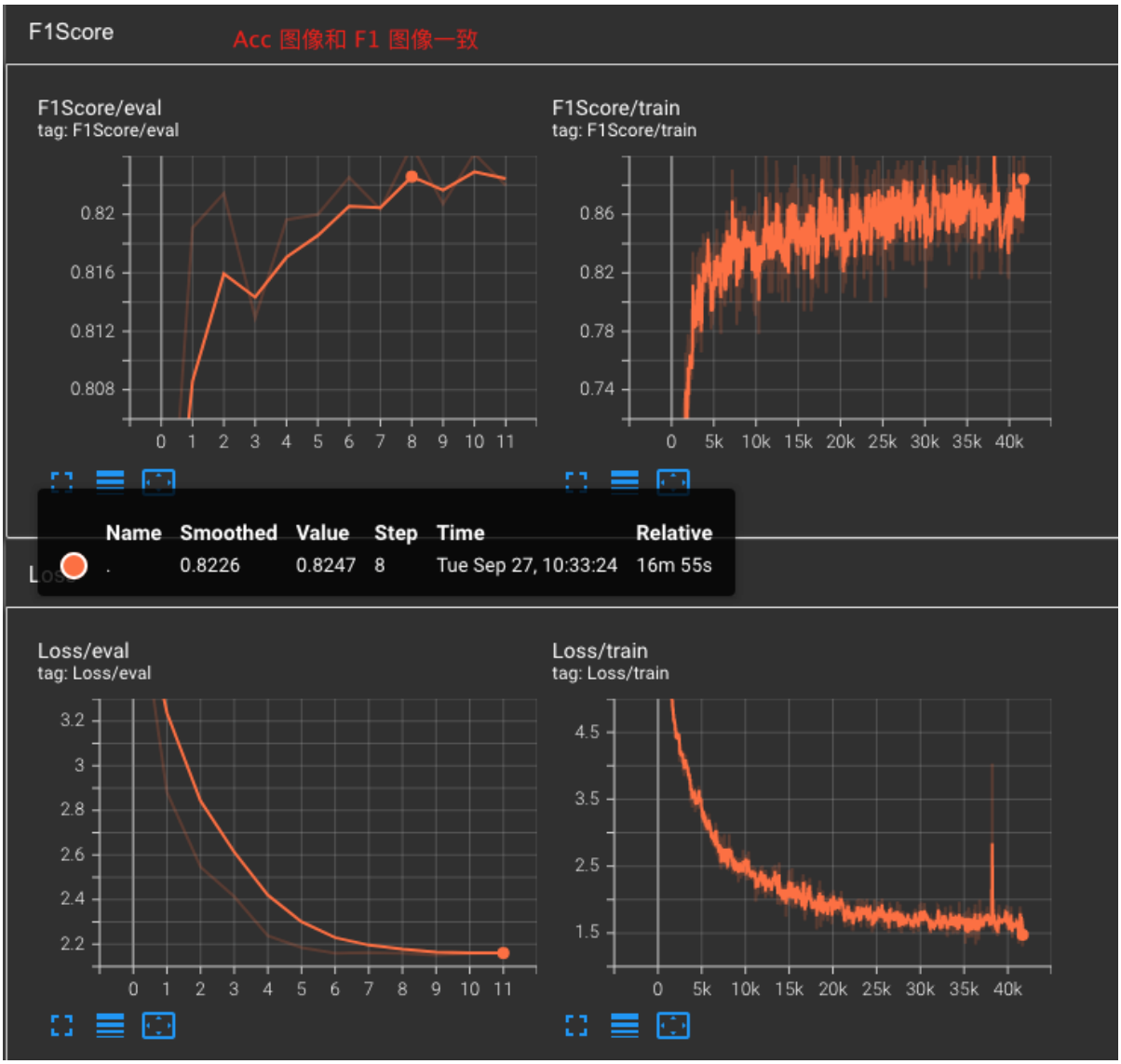


- Acc 均值，从 0.817 -> 0.8207，提升有限

- **实验二【更困难样本扩充】**

```
原始训练数据 104889 条
扩充更困难样本 6248 条
合计数据 111137 条
```

-

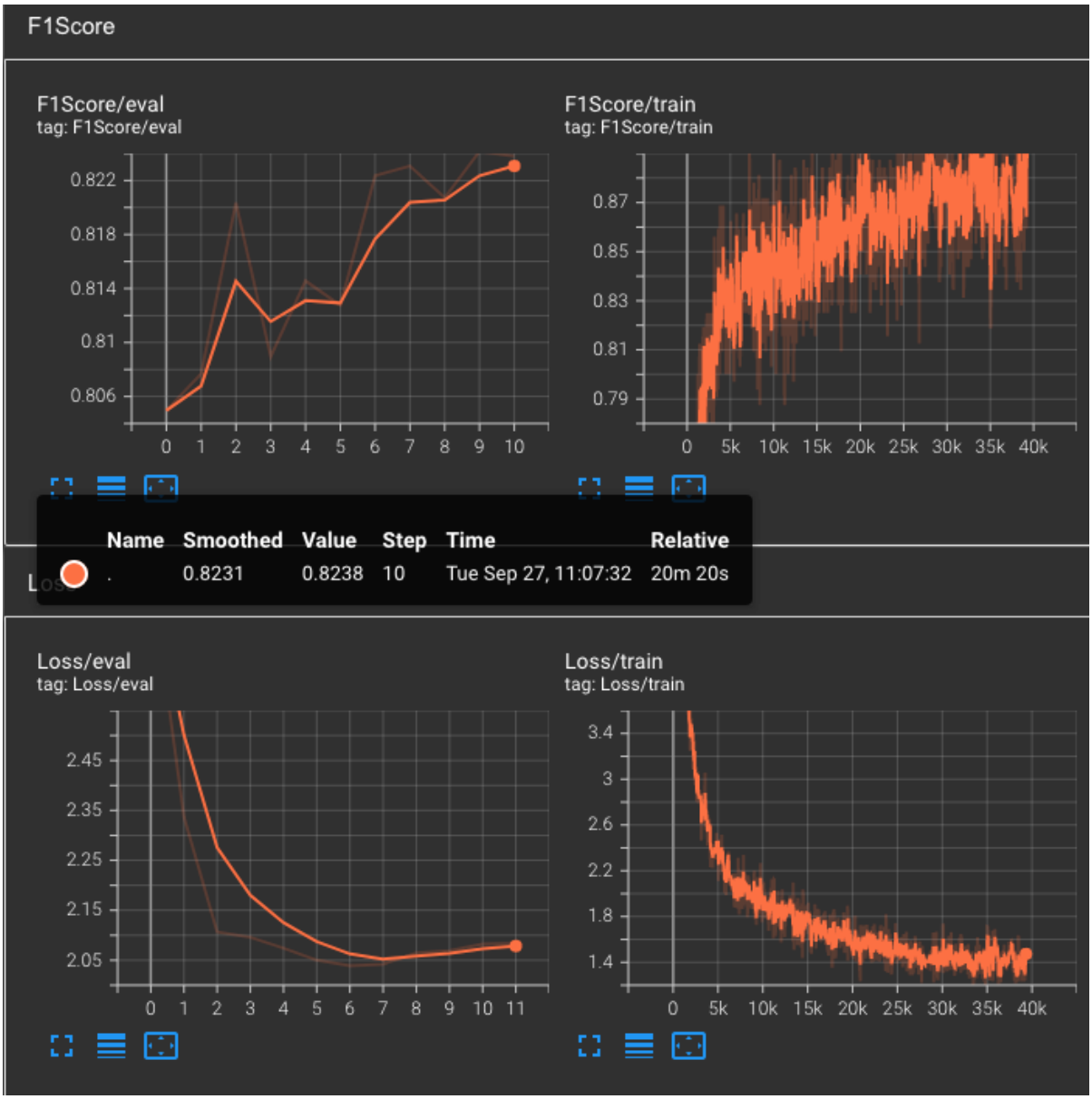


- Acc 均值，从 0.817 -> 0.8247，提升明显

- **实验三【重标样本修复】**

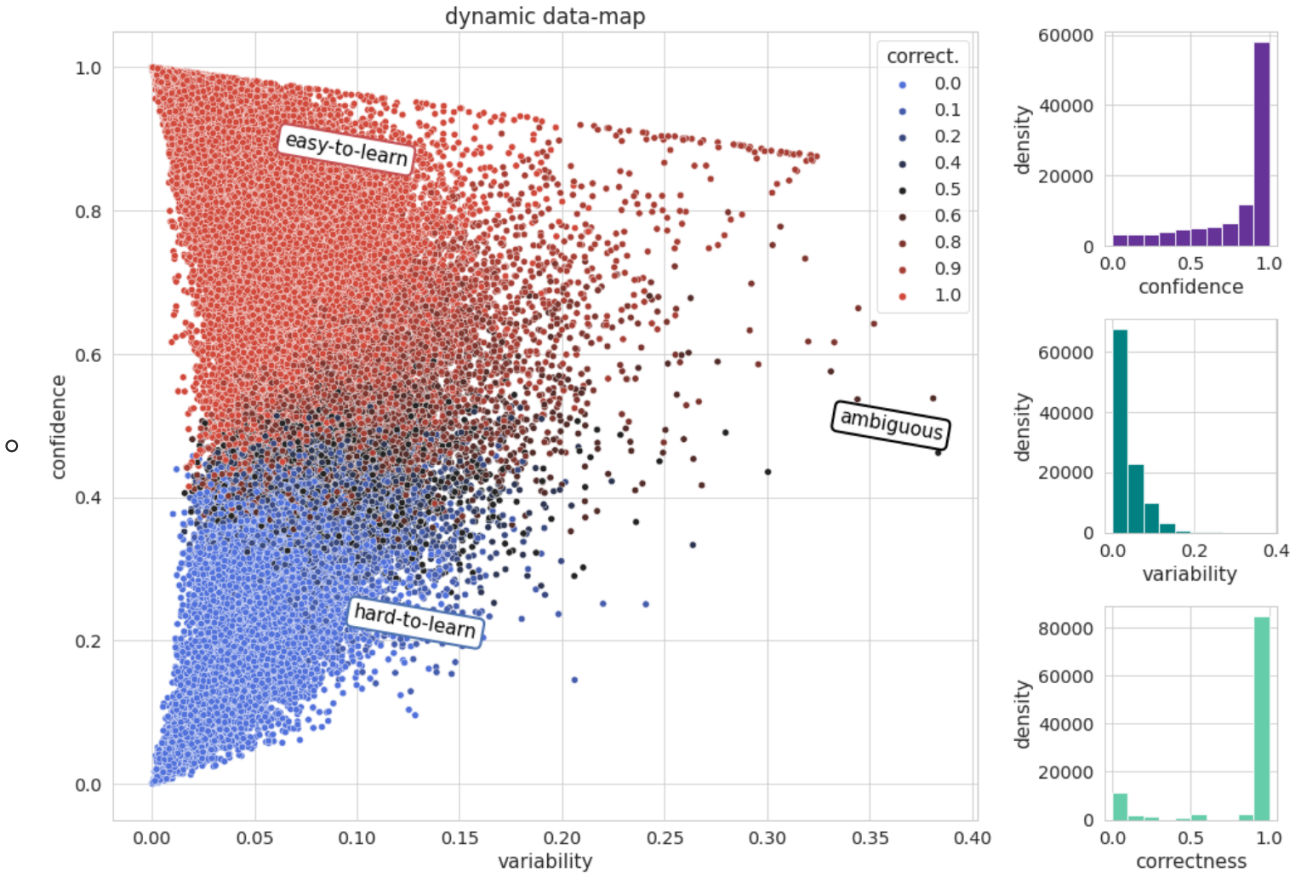
```
原始训练数据 104889 条
替换 review 数据 2971 条
```

-



    - Acc 均值，从 0.817 -> 0.8227，提升明显

- 根据上面的的数据方面改进，我们重新分析了训练数据



- 发现整体的方差更小，说明数据一致性相比改进前更好

四、讨论和结论

- 根据时效性项目的数据分析的实践，我们发现针对训练数据分析归类，并针对性地采取措施，对于提升模型效果，节约标注资源，是一个非常有效的手段
- 我们应该更加关注左下角的 Hard Case，针对这部分筛选错标数据和扩充数据，对于提升模型效果非常明显

五、拓展阅读

- 时效性相关
  - https://wiki.in.zhihu.com/pages/viewpage.action?pageId=357278965
  - https://wiki.in.zhihu.com/pages/viewpage.action?pageId=357278965&preview=/357278965/357279005/QuestionDataAnalysis.pdf
- 主动学习相关
  - https://wiki.in.zhihu.com/pages/viewpage.action?pageId=331730826
- 动态分析数据相关
  - https://arxiv.org/pdf/2009.10795.pdf
- 代码

```
import json
import logging
```

```
import os
import pathlib
from collections import defaultdict
from itertools import groupby

import numpy as np
import pandas as pd

from common.utils import get_path_name
from common.utils import zmath
from common.utils.zjson import NumpyEncoder

logger = logging.getLogger(__name__)


class Cartographer:
    """to record/display dynamic information

    ref:
      1. 2020-Dataset Cartography: Mapping and Diagnosing Datasets with
Training Dynamics
      2. 2020-Identifying Mislabeled Data using the Area Under the Margin
Ranking

    使用方法:
        1）训练时开启记录：在配置文件 xxx_config.yaml 里增加 dynamics 相关配置(可参照
torch_demo)
        2）训练结束后：数据记录在 /data/models/model_name/model_version/dynamics
下，然后利用 cartographer 加载、作图、筛数据即可
        Example:
        >>> from common.utils.cartographer import Cartographer
        >>> cart =
Cartographer('/data/models/model_name/model_version/dynamics.epoch.jsonl')
        >>> cart.load()
        >>> cart.plot(cart.dynamics)
        or:
        >>> # 取 confidence >= 0.95 的
        >>> dynamics = cart.query_from_dynamics(cart.dynamics,
confidence_min=0.95)
        >>> cart.plot(dynamics)

    """

    def __init__(self, filename):
        """

        Args:
            filename: filename for saving dynamics
        """
```

```
        path, _, _ = get_path_name(filename)
        pathlib.Path(path).mkdir(parents=True, exist_ok=True)

        self.filename = filename

        self.records = []
        self.accum_margins = defaultdict(float)
        self.sample_counts = defaultdict(int)

        # will be updated with self.load()
        self.dynamics = None
        self.aums = None

    def update(self, step, logits, targets, ids):
        """

        Args:
            step:
            logits: (numpy array) logit of each sample in current batch, shape
is (batch_size, class_nums)
            targets: (numpy array) target of each sample in current batch,
shape is (batch_size, )
            ids: (numpy array) id of each sample in current batch, shape is
(batch_size, )
        """
        n_classes = logits.shape[1]

        # 对 targets 进行 one-hot 化
        mask = np.squeeze(np.eye(n_classes)[targets])
        # 取出 targets 对应的 logits 分数, shape is (batch_size, )
        target_logits = np.max(np.where(mask > 0, logits,
np.zeros_like(logits)), axis=1)

        mask_inv = 1 - mask
        # 取出非 targets 对应的最大 logits 分数, shape is (batch_size, )
        other_logits_max = np.max(np.where(mask_inv > 0, logits,
np.zeros_like(logits)), axis=1)

        # 如果训练不好的话，存在负数，一般认为模型的收敛能力还行，负数比例不会超过 20%，如果
发现负数很多，说明模型没有收敛
        margins = target_logits - other_logits_max
        predicts = np.argmax(logits, axis=1)

        for idx, margin in enumerate(margins):
            self.sample_counts[ids[idx]] += 1
            # 这里为啥不用绝对值，而是直接加呢？因为还是认为模型收敛还行，负数不多，直接拉平
就行；用绝对值应该也行
            self.accum_margins[ids[idx]] += margin
```

```
            # pred_prob_gold 是 softmax 之后，target 对应的预测概率
            pred_prob_gold = zmath.softmax(logits[idx]).tolist()[0]
[int(targets[idx])]
            pred_is_correct = 1 if np.argmax(logits[idx]) == targets[idx] else
0

            record = {
                'gid': step,
                'sid': ids[idx],
                'logits': logits[idx],
                'gold': targets[idx],
                'pred': predicts[idx],
                'pred_prob_gold': pred_prob_gold,
                'pred_is_correct': pred_is_correct,
                'margin': margin,
                'aum': self.accum_margins[ids[idx]] /
self.sample_counts[ids[idx]]
            }

            self.records.append(record)

    def finalize(self):
        logger.info(f'save raw dynamics into {self.filename}')

        correctness_ = {}
        confidence_ = {}
        variability_ = {}
        forgetfulness_ = {}

        records = sorted([
            {
                'gid': record['gid'],
                'sid': record['sid'],
                'pred_prob_gold': record['pred_prob_gold'],
                'pred_is_correct': record['pred_is_correct']
            } for record in self.records
        ], key=lambda record: record['sid'])

        for sid, group in groupby(records, key=lambda record: record['sid']):
            info = [g for g in group]
            true_prob_trends = sorted([(g['gid'], g['pred_prob_gold']) for g in
info], key=lambda g: g[0])
            true_prob_trends = [prob for _, prob in true_prob_trends]

            correctness_trends = sorted([(g['gid'], g['pred_is_correct']) for g
in info], key=lambda g: g[0])
            correctness_trends = [cc for _, cc in correctness_trends]

            correctness_[sid] = self.fn_correctness(correctness_trends)
```

```
            confidence_[sid] = self.fn_confidence(true_prob_trends)
            variability_[sid] = self.fn_variability(true_prob_trends)

            forgetfulness_[sid] = self.fn_forgetfulness(correctness_trends)

        with open(self.filename, 'w') as fp:
            for record in self.records:
                sid = record['sid']

                record['correctness'] = correctness_[sid]
                record['confidence'] = confidence_[sid]
                record['variability'] = variability_[sid]
                record['forgetfulness'] = forgetfulness_[sid]

                fp.write(json.dumps(record, cls=NumpyEncoder) + '\n')
        logger.info('all raw dynamics are saved')

    @staticmethod
    def fn_correctness(correctness_trends):
        return sum(correctness_trends)

    @staticmethod
    def fn_confidence(true_prob_trends):
        return np.mean(true_prob_trends)

    @staticmethod
    def fn_variability(true_prob_trends):
        return np.std(true_prob_trends)

    @staticmethod
    def fn_forgetfulness(correctness_trends):
        if not any(correctness_trends):  # never learnt
            return 10000

        learnt = False
        times_forgotten = 0
        for is_correct in correctness_trends:
            if (not learnt and not is_correct) or (learnt and is_correct):
                continue
            elif learnt and not is_correct:
                learnt = False
                times_forgotten += 1
            elif not learnt and is_correct:
                learnt = True

        return times_forgotten

    def load(self):
        logger.info(f'loading raw dynamics from {self.filename}')
```

```
        js = pd.read_json(path_or_buf=self.filename, lines=True)
        self.records = js.to_dict('records')

        self.dynamics = js[
            ['sid', 'correctness', 'confidence', 'variability',
'forgetfulness']
        ].drop_duplicates(subset=['sid']).assign(corr_frac=lambda d:
d.correctness / d.correctness.max())

        self.aums = js[['gid', 'sid', 'aum']].drop_duplicates(subset=['gid',
'sid'])

    @staticmethod
    def plot(dynamics, filename=None):
        """plot confidence-variability and save it into png"""
        import matplotlib.pyplot as plt
        import seaborn as sns

        sns.set(style='whitegrid', font_scale=1.6, font='Georgia',
context='paper')
        data = dynamics
        data['correct.'] = [float(f'{x:.2f}') for x in data['corr_frac']]

        metric_variability = 'variability'
        metric_confidence = 'confidence'
        metric_correct_frac = 'correct.'

        num_correct_frac = len(data[metric_correct_frac].unique().tolist())
        style = metric_correct_frac if num_correct_frac < 8 else None

        fig = plt.figure(figsize=(14, 10), )
        gs = fig.add_gridspec(3, 2, width_ratios=[5, 1])
        ax_confidence_variability = fig.add_subplot(gs[:, 0])

        pal = sns.diverging_palette(260, 15, n=num_correct_frac, sep=10,
center='dark')
        plot = sns.scatterplot(
            x=metric_variability, y=metric_confidence,
ax=ax_confidence_variability,
            data=data, hue=metric_correct_frac, palette=pal, style=style, s=30
        )

        # annotate Regions
        def fn_bbox(c):
            return dict(boxstyle='round,pad=0.3', ec=c, lw=2, fc='white')

        def fn_annotate(text, xyc, bbc):
            return ax_confidence_variability.annotate(
                text,
```

```
                xy=xyc,
                xycoords='axes fraction',
                fontsize=15,
                color='black',
                va='center',
                ha='center',
                rotation=350,
                bbox=fn_bbox(bbc)
            )

        fn_annotate('ambiguous', xyc=(0.9, 0.5), bbc='black')
        fn_annotate('easy-to-learn', xyc=(0.27, 0.85), bbc='r')
        fn_annotate('hard-to-learn', xyc=(0.35, 0.25), bbc='b')

        plot.set_xlabel('variability')
        plot.set_ylabel('confidence')
        plot.set_title('dynamic data-map', fontsize=17)

        # plot the histograms
        ax_hist_confidence = fig.add_subplot(gs[0, 1])
        hist_confidence = data.hist(column=['confidence'],
ax=ax_hist_confidence, color='RebeccaPurple')
        hist_confidence[0].set_title('')
        hist_confidence[0].set_xlabel('confidence')
        hist_confidence[0].set_ylabel('density')

        ax_hist_variability = fig.add_subplot(gs[1, 1])
        hist_variability = data.hist(column=['variability'],
ax=ax_hist_variability, color='Teal')
        hist_variability[0].set_title('')
        hist_variability[0].set_xlabel('variability')
        hist_variability[0].set_ylabel('density')

        ax_hist_correctness = fig.add_subplot(gs[2, 1])
        hist_correctness = data.hist(column=['correct.'],
ax=ax_hist_correctness, color='MediumAquaMarine')
        hist_correctness[0].set_title('')
        hist_correctness[0].set_xlabel('correctness')
        hist_correctness[0].set_ylabel('density')

        fig.tight_layout()

        if filename is not None:
            filename_figure = f'{os.path.splitext(filename)[0]}.png'
            logger.info(f'save figure into {filename_figure}')
            fig.savefig(filename_figure, dpi=300)

    @staticmethod
    def query_from_dynamics(
```

```
        dynamics,
        confidence_min=None, confidence_max=None,
        variability_min=None, variability_max=None,
        corr_frac_min=None, corr_frac_max=None,
        forgetfulness_min=None, forgetfulness_max=None
    ):
        """query data which satisfy conditions:
            [confidence_min, confidence_max)
            [variability_min, variability_max)
            [corr_frac_min, corr_frac_max)
            [forgetfulness_min, forgetfulness_max)
        """
        def _is_valid_threshold(theta):
            return theta is not None and isinstance(theta, float) and 0.0 <=
theta <= 1.0

        left_thresholds = [
            ('confidence', confidence_min),
            ('variability', variability_min),
            ('corr_frac', corr_frac_min),
            ('forgetfulness', forgetfulness_min)
        ]
        left_thresholds = [(k, v) for k, v in left_thresholds if
_is_valid_threshold(v)]

        right_thresholds = [
            ('confidence', confidence_max),
            ('variability', variability_max),
            ('corr_frac', corr_frac_max),
            ('forgetfulness', forgetfulness_max)
        ]
        right_thresholds = [(k, v) for k, v in right_thresholds if
_is_valid_threshold(v)]

        selected_idx = dynamics['confidence'] >= dynamics['confidence'].min()
        for k, v in left_thresholds:
            selected_idx &= dynamics[k] >= v

        for k, v in right_thresholds:
            selected_idx &= dynamics[k] < v

        return dynamics[selected_idx]
```