**1 官方版能力概览**



**2 个人理解**

**2.1 证据分析-基于特征和基于实例**

☐ 下载TrustAI库和InterpretDL库

<https://github.com/PaddlePaddle/TrustAI/tree/data_map/trustai/interpretation>

**2.1.1 特征级证据分析**

示例：[基于TrustAI的特征级证据分析示例-中文情感分析任务](https://aistudio.baidu.com/aistudio/projectdetail/4431334)

准备训练好的模型

待测数据

特征分析算法接口

计算得到原模型预测结果，类别预测概率，特征重要度分数

计算分词的重要性

**导入包**

import numpy as np

import paddle

from paddlenlp.transformers import ErnieForSequenceClassification, ErnieTokenizer

from paddlenlp.datasets import load\_dataset

from assets.utils import training\_model

import os

from assets.utils import training\_model

from assets.utils import preprocess\_fn

from trustai.interpretation.token\_level import IntGradInterpreter

from trustai.interpretation.token\_level.common import attention\_predict\_fn\_on\_paddlenlp

from trustai.interpretation.token\_level import AttentionInterpreter

from trustai.interpretation.token\_level import LIMEInterpreter

from trustai.interpretation.token\_level import GradShapInterpreter

from trustai.interpretation import VisualizationTextRecord

from trustai.interpretation import visualize\_text

import jieba

from trustai.interpretation import get\_word\_offset

**加载模型，加载数据**

# Select pre-trained model

MODEL\_NAME = "ernie-3.0-base-zh" # choose from ["ernie-1.0", "ernie-1.0-base-zh", "ernie-1.0-large-zh-cw", "ernie-2.0-base-zh", "ernie-2.0-large-zh", "ernie-3.0-xbase-zh", "ernie-3.0-base-zh", "ernie-3.0-medium-zh", "ernie-3.0-mini-zh", "ernie-3.0-micro-zh", "ernie-3.0-nano-zh"]

# Select dataset for model training

DATASET\_NAME = 'chnsenticorp'

# Set the path to save the trained model

MODEL\_SAVE\_PATH = f'./save\_model/{DATASET\_NAME}-{MODEL\_NAME}'

# Init model and tokenizer

model = ErnieForSequenceClassification.from\_pretrained(MODEL\_NAME, num\_classes=2)

tokenizer = ErnieTokenizer.from\_pretrained(MODEL\_NAME)

# Load dataset

train\_ds, dev\_ds = load\_dataset(DATASET\_NAME, splits=["train", "dev"])

# 重新训练一个中文情感分析模型

# training\_model(model, tokenizer, train\_ds, dev\_ds, save\_dir=MODEL\_SAVE\_PATH)

# 加载已训练好模型的参数

state\_dict = paddle.load(os.path.join(MODEL\_SAVE\_PATH, "model\_state.pdparams"))

model.set\_dict(state\_dict)

data = [ {"text":""}]

model\_input = preprocess\_fn(data, tokenizer)

# model\_input是模型的输入，格式必须满足`logits = model(\*model\_input)`。

# 模型的输出格式必须包含且仅包含`logits`，若不满足这种格式（如模型的输出为`logit，probs = model(\*model\_input)`），需要用户自定义predict\_fn，相关细节见后面的内容。

**特征级解释**

**IG interpreter**

# Init an IG interpreter

# 指定gpu device为卡0

ig\_interpreter = IntGradInterpreter(model, device="gpu:0")

# Use IG interpreter to get the importance scores for all data

ig\_results = ig\_interpreter(model\_input)

ig\_results[0]

# ig\_results: List[IGResult]，len(ig\_results)等于测试样本的数量

# IGResult.attributions: List[float]， 测试数据中每一个特征（字或词）的重要度分数（由于model\_input包含多条测试样本，因此len(ig\_results[0].attributions)中包含padding长度）

# IGResult.pred\_label: float， 测试样本预测label

# IGResult.pred\_proba: List[float]，测试样本的在每个类别的预测概率

**attention interpreter**

# Init an attention interpreter and get the importance scores

# 每个模型计算attention的方式都不相同，因此用户可以参照`attention\_predict\_fn\_on\_paddlenlp`实现对应自己模型的`predict\_fn`"gpu:0", predict\_fn=attention\_predict\_fn\_on\_paddlenlp)

att\_interpreter = AttentionInterpreter(model, device="gpu:0", predict\_fn=attention\_predict\_fn\_on\_paddlenlp)

**LIME interpreter**

# Hyperparameters

LIME\_SAMPLES = 1000

# Init an LIME interpreter

# LIMEInterpreter需要用户指定unk和pad字符在词表中的索引

lime\_interpreter = LIMEInterpreter(model, device="gpu:0",

unk\_id=tokenizer.convert\_tokens\_to\_ids('[UNK]'),

pad\_id=tokenizer.convert\_tokens\_to\_ids('[PAD]'))

**GradShap interpreter**

# Init an GradShap interpreter

# gradshap的device参数仅支持输入`gpu`和`cpu`

grad\_interpreter = GradShapInterpreter(model, device='gpu')

**单字可视化**

# 将id转为为token

tokens = [tokenizer.convert\_ids\_to\_tokens(\_input\_ids) for \_input\_ids in model\_input[0].tolist()]

# 去除[PAD]字符

tokens = [[token for token in \_tokens if token != '[PAD]'] for \_tokens in tokens]

true\_labels = [0, 0, 1]

# IG方法的可视化结果

recs = []

for i in range(len(ig\_results)):

recs.append(VisualizationTextRecord(lime\_results[i], words=tokens[i], true\_label=true\_labels[i]))

html = visualize\_text(recs)

**关键词可视化**

# Add CLS and SEP tags to both original text and standard splited tokens

contexts = []

words = []

for exmaple in data:

text = exmaple['text']

# 由于本示例使用的ERNIE模型，因此需要在文本前后分别补充"[CLS]"和"[SEP]"两个字符，才能与模型输出结果对齐。

# context示例：[CLS]这个宾馆比较陈旧了，特价的房间也很一般。总体来说一般[SEP]

contexts.append("[CLS]" + text + "[SEP]")

# 相比模型输入，更大粒度的切词，此处使用jieba的切词。

# word示例：['[CLS]', '这个', '宾馆', '比较', '陈旧', '了', '，', '特价', '的', '房间', '也', '很', '一般', '。', '总体', '来说', '一般', '[SEP]']

words.append(["[CLS]"] + list(jieba.cut(text)) + ["[SEP]"])

# Get the offset map of tokenized tokens and standard splited tokens

ori\_offset\_maps = []

word\_offset\_maps = []

for i in range(len(contexts)):

ori\_offset\_maps.append(tokenizer.get\_offset\_mapping(contexts[i]))

word\_offset\_maps.append(get\_word\_offset(contexts[i], words[i]))

align\_res = lime\_interpreter.alignment(lime\_results, contexts, words, word\_offset\_maps, ori\_offset\_maps, special\_tokens=["[CLS]", '[SEP]'])

# process for vbisualize

recs = []

for i in range(len(align\_res)):

recs.append(VisualizationTextRecord(align\_res[i], true\_label=true\_labels[i]))

html = visualize\_text(recs)

更多方法：[特征级证据分析文档](https://github.com/PaddlePaddle/TrustAI/blob/main/trustai/interpretation/token_level/README.md)

**重点重要度分数**

import jieba

from trustai.interpretation import get\_word\_offset

# 分析的文本

print(data[0]['text']）

# 这个宾馆比较陈旧了，特价的房间也很一般。总体来说一般

# 由于示例使用的是ernie-1.0模型，若要与attribution对齐，需要在原始文本拼接[CLS], [SEP]

context = "[CLS]" + " " + data[0]['text'] + " " + "[SEP]"

# 开发者自定分词

words = ["[CLS]"] + list(jieba.cut(data[0]['text'])) + ["[SEP]"]

# ['[CLS]', '这个', '宾馆', '比较', '陈旧', '了', '，', '特价', '的', '房间', '也', '很', '一般', '。', '总体', '来说', '一般', '[SEP]']

# 获取用户自定切词的word\_offset\_map

# word\_offset\_map表示开发者自定义切词在context中的字符的偏移位置

word\_offset\_map = get\_word\_offset(context, words)

# [[0, 5], [6, 8], [8, 10], [10, 12], [12, 14], [14, 15], [15, 16], [16, 18], [18, 19], [19, 21], [21, 22], [22, 23], [23, 25], [25, 26], [26, 28], [28, 30], [30, 32], [33, 38]]

# 计算模型切词offset\_map

subword\_offset\_map = tokenizer.get\_offset\_mapping(context)

# [(0, 1), (1, 3), (3, 4), (4, 5), (6, 7), (7, 8), (8, 9), (9, 10), (10, 11), (11, 12), (12, 13), (13, 14), (14, 15), (15, 16), (16, 17), (17, 18), (18, 19), (19, 20), (20, 21), (21, 22), (22, 23), (23, 24), (24, 25), (25, 26), (26, 27), (27, 28), (28, 29), (29, 30), (30, 31), (31, 32), (33, 34), (34, 37), (37, 38)]

# 将attributions对齐到words

aligns = ig.alignment(result, [context], [batch\_words], [word\_offset\_map], [subword\_offset\_map], special\_tokens=["[CLS]", '[SEP]'])

print(aligns[0].words)

# ['[CLS]', '这个', '宾馆', '比较', '陈旧', '了', '，', '特价', '的', '房间', '也', '很', '一般', '。', '总体', '来说', '一般', '[SEP]']

print(aligns[0].word\_attributions)

# [0.021498650312423706, 0.17480190843343735, 0.3245609328150749, 0.013578088022768497, 0.02439146302640438, -0.01771647110581398, 0.05467343330383301, 0.024022094905376434, 0.020944949239492416, -0.012377424165606499, 0.03983632102608681, 0.05040765926241875, 0.14022761583328247, -0.024755112826824188, -0.13763408362865448, 0.01691947504878044, 0.001623895950615406, 0.015423357486724854]

print(aligns[0].pred\_label)

# 0

print(aligns[0].pred\_proba)

# [0.86797816 0.1320218 ]

print(aligns[0].rationale)

# (1, 2, 6, 11, 12)

print(aligns[0].rationale\_tokens)

# ('这个', '宾馆', '，', '很', '一般')

print(aligns[0].non\_rationale)

# (3, 4, 5, 7, 8, 9, 10, 13, 14, 15, 16)

print(aligns[0].non\_rationale\_tokens)

# ('比较', '陈旧', '了', '特价', '的', '房间', '也', '。', '总体', '来说', '一般')

**2.1.2 实例级证据分析**

示例：[基于TrustAI的实例级证据分析示例-中文情感分析任务](https://aistudio.baidu.com/aistudio/projectdetail/4433286)

从训练数据中找出对当前预测文本影响较大的若干训练样本作为模型预测依赖证据

要加载所有的训练集数据

加载原模型

输入测试数据

train\_ds, dev\_ds = load\_dataset(DATASET\_NAME, splits=["train", "dev"])

train\_dataloader = create\_dataloader\_from\_scratch(train\_ds.data, tokenizer, batch\_size=16)

**FeatureSimilarityModel**

from trustai.interpretation import FeatureSimilarityModel

# init interpreter

# classifier\_layer\_name: 模型最后一层全连接层的layer name, 用户需要手动指定， 默认为'classifier'

interpreter = FeatureSimilarityModel(model, train\_dataloader, classifier\_layer\_name='classifier')

# sim\_fn：计算相似度的函数，可从`cos`, `dot`, `euc`中任选其一

# sample\_num：将训练数据按照影响分数排序后，分别返回的正影响和负影响证据的数量，设置为-1时返回排序后的全部训练数据。

fs\_result = interpreter(model\_input, sample\_num=3, sim\_fn="cos")

# fs\_result: List[ExampleResult], len(fs\_result)等于测试样本的数量

# ExampleResult.pos\_indexes: List[int], 正影响证据（支持模型预测）在训练集中的索引。

# ExampleResult.neg\_indexes: List[int], 负影响证据（不支持模型预测）在训练集中的索引。

# ExampleResult.pos\_scores: List[float], 正影响证据的证据分数

# ExampleResult.neg\_scores: List[float], 负影响证据的证据分数

**RepresenterPointModel**

from trustai.interpretation import RepresenterPointModel

# init interpreter

# classifier\_layer\_name: 模型最后一层全连接层的layer name, 用户需要手动指定， 默认为'classifier'

interpreter = RepresenterPointModel(model, train\_dataloader, classifier\_layer\_name='classifier')

# 结果形式与fs\_result相同

# sample\_num：将训练数据按照影响分数排序后，分别返回的正影响和负影响证据的数量，设置为-1时返回排序后的全部训练数据。

rp\_result = interpreter(model\_input, sample\_num=3)

# 格式化打印结果

print\_result(data, train\_ds, rp\_result, data\_name='chnsenticorp')

**GradientSimilarityModel**

from trustai.interpretation import GradientSimilarityModel

# GradientSimilarity方法传入的训练数据和测试数据的batch\_size都必须为1

# 满足`logits = model(\*next(train\_data\_loader))`

train\_dataloader = create\_dataloader\_from\_scratch(train\_ds.data, tokenizer, batch\_size=1)

model\_inputs = create\_dataloader\_from\_scratch(data, tokenizer, batch\_size=1)

# init interpreter

interpreter = GradientSimilarityModel(model, train\_dataloader, classifier\_layer\_name='classifier')

gs\_result = []

for model\_input in model\_inputs:

# 结果形式与fs\_result相同

# sim\_fn：计算相似度的函数，可从`cos`, `dot`中任选其一

# sample\_num：将训练数据按照影响分数排序后，分别返回的正影响和负影响证据的数量，设置为-1时返回排序后的全部训练数据。

gs\_result += interpreter(model\_input, sample\_num=3, sim\_fn="cos")

# 格式化打印结果

print\_result(data, train\_ds, gs\_result, data\_name='chnsenticorp')

更多方法：[实例级证据分析文档](https://github.com/PaddlePaddle/TrustAI/blob/main/trustai/interpretation/example_level/README.md)

基于梯度方法：基于训练样本的梯度计算其对模型的影响度，再通过影响度计算训练样本和测试样本的表示，以及两者的分数；一般可以根据这种方法得到脏数据

基于表示相似度方法：计算训练样本i和测试样本t的表示的相似度

基于梯度相似度方法：计算训练样本i和测试样本t的梯度的相似度

**2.2 数据分析-标注错误、标注不足、分布偏置**

**个人经验**

标注错误方法：筛选训练数据中相似度高 但标注结果不一致的数据，人工校核错误数据

标注不足：统计训练数据中关键词与标签的分布，测试数据中关键词与标签的分布，对比查看

偏置没有解决

识别上述问题

<https://github.com/PaddlePaddle/TrustAI/tree/data_map/tutorials>

**2.2.1 数据标注错误识别**

 [训练数据中标注错误数据自动识别示例](https://aistudio.baidu.com/aistudio/projectdetail/4434058)

脏数据选择方法为：使用TrustAI提供的实例级可信分析方法RepresenterPointModel，计算训练集中样本对模型loss的影响分数，分数越大表明样本为脏数据的可能性越大，模型在这些样本上表现也相对较差。

tutorials/dirty\_data\_identification/find\_dirty\_data.py

**2.2.2 数据覆盖不足识别**

[训练数据覆盖不足识别示例](https://aistudio.baidu.com/aistudio/projectdetail/4434403)

目标集选择方法为：使用TrustAI提供的实例级可信分析FeatureSimilarityModel方法，计算验证集中样本的正影响证据的平均分数。分数较低的样本表明其训练证据不足，训练数据对此类数据覆盖度较低，模型在这些样本上表现也相对较差。

tutorials/sparse\_data\_identification/find\_sparse\_data.py

得到可能的稀疏数据后，再使用FeatureSimilarityModel从未标注的数据集中选择相似的数据进行人工标注

tutorials/sparse\_data\_identification/find\_valid\_data.py

标注后合并至原训练集中，重新训练

**2.2.3 数据分布偏置识别**

神经网络模型会利用数据集中的偏置作为预测捷径，如在情感分析任务中，遇到否定词模型会倾向预测为“负向”情感。这种偏置会导致模型没有真正理解语言，导致模型的鲁棒性降低。

[偏置识别](https://github.com/PaddlePaddle/TrustAI/tree/data_map/tutorials/data_bias_identification)

**2.2.3.1 基于统计的偏置识别**

**2.2.3.1.1 数据权重修正**

<https://github.com/PaddlePaddle/TrustAI/tree/data_map/tutorials/data_bias_identification/less_learn_shortcut>

更少学习捷径:伪特征-标签相关性的分析与缓解学习：Less Learn Shortcut: Analyzing and Mitigating Learning of Spurious Feature-Label Correlation

降低偏置样本对训练loss的影响，即减少模型从偏置样本中学习

偏置识别：统计训练数据中词与标注标签的分布，在分布上不均衡的词可能是偏置词，包含偏置词的样本为偏置样本，计算词出现的次数，属于某个类别占比高则认为偏置 (源码中只输出了偏置词的占比,没有输出对应的标签)

权重修正：降低偏置样本对训练loss的影响，即针对每一条样本计算一个偏置度，在训练loss计算时通过偏置度降低偏置样本影响

示例：<https://aistudio.baidu.com/aistudio/projectdetail/4434616>

tutorials/data\_bias\_identification/less\_learn\_shortcut

**2.2.3.1.2 数据分布修正**

<https://github.com/PaddlePaddle/TrustAI/tree/data_map/tutorials/data_bias_identification/data_distribution_correction>

通过对非偏置数据多次重复采样，使训练数据分布尽量均衡

偏置识别：统计训练数据中词与标注标签的分布，在分布上不均衡的词可能是偏置词，这里需要使用任务相关词典对候选偏置词过滤，得到真正的偏置词。包含偏置词的样本为偏置样本

分布修正：对非偏置样本进行重复采样

首先，通过可信分析识别训练数据中对模型预测其重要贡献的证据。 重要证据统计方法为：基于特征级可信分析方法IntGradInterpreter识别训练数据中起重要贡献的证据和频次。

**2.2.3.2 基于模型的偏置识别**

**2.3 可信增强**

对数据分析的结果，针对性的进行处理

上述数据分析过程已经处理了标注不足和数据分布偏置的问题

**2.3.1 证据识别及基于证据的预测**

[解决文本冗余导致精度下降的问题](https://github.com/PaddlePaddle/TrustAI/blob/data_map/tutorials/redundancy_removal)

针对长文本理解，排除冗余数据

**2.3.2 基于证据指导的模型增强方案**

[基于证据指导的模型增强方案](https://github.com/PaddlePaddle/TrustAI/blob/data_map/tutorials/enhanced_by_rationale)

即标注少量证据数据，通过联合学习原始任务和证据学习任务，用证据学习目标指导模型依赖合理的证据进行预测，提升模型可解释性

**2.3.3 基于证据指导的预测错误数据识别**

[相似度计算任务的预测错误数据识别](https://github.com/PaddlePaddle/TrustAI/blob/data_map/tutorials/map_analysis/zh-similarity-application.ipynb)

对模型预测依赖证据的分析，识别潜在预测错误的数据；利用TrustAI提供的特征级证据分析方法识别模型预测依赖证据

假设：预测依赖证据合理性弱，说明预测结果不可信

怎么表示证据合理性弱？通过对比预测出的相似句子 各自的特征级证据是否高度一致 来判断。-（这个可以借鉴使用）

**参考资料**

TrustAI-Github：<https://github.com/PaddlePaddle/TrustAI>

推广TrustAI可信分析：通过提升数据质量来增强在ERNIE模型下性能 <https://aistudio.baidu.com/aistudio/projectdetail/4622139?contributionType=1>

百度重磅发布业界首个集可信分析和增强于一体的可信AI工具集-TrustAI <https://developer.baidu.com/article/detail.html?id=295219>

<https://juejin.cn/post/7158638835297943566>

**阅读次数**

☐ 2023/02/02 15:48 -2023/02/02 19:29