Kaggle - Detect AI Generated Text

赛后讲解

主讲人：H老师

赛题背景

竞赛概览

在近年来，大语言模型（LLM）的发展日益成熟，它们生成的文本越来越难以与人类写作区分。本次竞赛旨在推动对实际应用中AI检测技术的开放研究和透明度。竞赛挑战参与者开发一个能准确检测出一篇文章是由学生还是LLM写成的机器学习模型。竞赛数据集包含了学生写的论文和由各种LLMs生成的论文。

这次竞赛的目标是构建一个机器学习模型，帮助识别哪些文章是学生写的，哪些是通过大语言模型撰写的。随着LLMs的普及，许多人担心它们将取代或改变通常由人类完成的工作。教育工作者尤其关注它们对学生技能发展的影响，尽管许多人保持乐观，认为LLMs最终将成为帮助学生提高写作技能的有用工具。本次竞赛希望通过识别LLM特有的迹象，推动LLM文本检测技术的发展，使用中等长度的文本和多个未知的生成模型，在各种主题上复现典型的检测场景，激励跨模型的特征学习。此次活动是由范德堡大学和位于亚利桑那州的独立非营利机构学习代理实验室与Kaggle合作举办的。

赛题背景

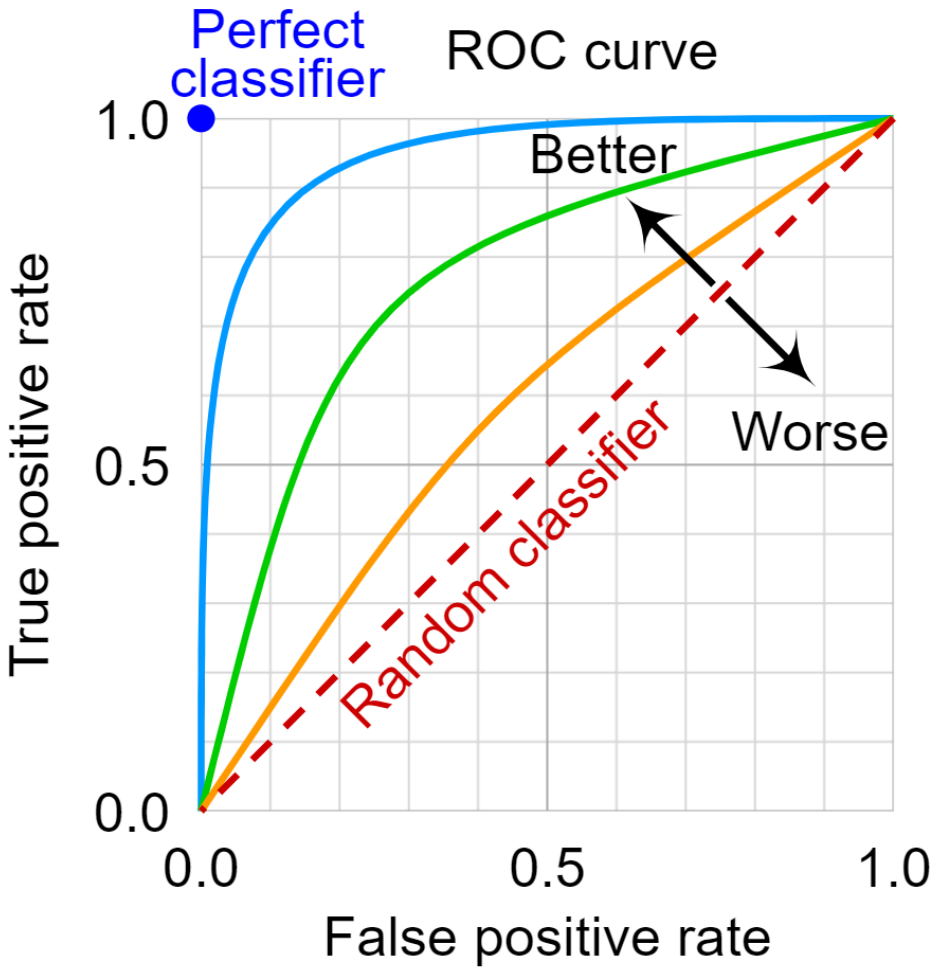
评价指标：AUC （area under the ROC ）

ROC曲线是一个用于评估二分类模型性能的图形化工具，它通过将真正例率（True Positive Rate, TPR）和假正例率（False Positive Rate, FPR）在不同阈值设置下的表现画出来进行评估。

AUC值的范围从0到1，一个完美的模型的AUC值为1，表示模型具有完美的区分能力，能够将所有的正例和负例完全区分开来。AUC值为0.5时，表示模型没有任何区分能力，等同于随机猜测。一般来说，AUC值越接近1，模型的性能越好；AUC值越低，模型的性能越差。

AUC作为一个评价指标，有几个显著的优点：

- 不依赖于特定的分类阈值：AUC考虑了所有可能的分类阈值，提供了一个模型性能的整体度量，这使得它非常适合用于评估和比较不同的模型。
- 对样本不平衡问题不敏感：在实际问题中，正负样本往往存在数量上的不平衡，而AUC作为性能度量指标，对这种不平衡不太敏感。
- 直观：AUC提供了一个直观的指标来衡量模型的预测能力，便于理解和解释。



数据概览

数据概览

竞赛官方只提供了1378个样本，其中由LLM生成的议论文样本**只有3个**，剩下全部是学生写作的议论文样本，右边是示例

LLM

```
train[train['generated']==1]['text'].iloc[0]
```

```
"This essay will analyze, discuss and prove one reason in favor of keeping the Electoral College in the United States for its pr
esidential elections. One of the reasons to keep the electoral college is that it is better for smaller, more rural states to ha
ve more influence as opposed to larger metropolitan areas that have large populations. The electors from these states are grante
d two votes each. Those from larger, more populated areas are granted just one vote each. Smaller states tend to hold significan
t power because their two votes for president and vice president add up more than the votes of larger states that have many elec
tors. This is because of the split of the electoral votes. Some argue that electors are not bound to vote for the candidate who
won the most votes nationally. They do not have to vote for their own state's nominee unless their state has a winner take all s
ystem. However, there are states that have adopted laws that force their electors to vote for their state's candidate. It seems
that, no matter how, electors are not bound to vote for the candidate who won the most nationally. This is not always the case b
ecause of state legislatures who can overrule the electors and vote for the alternative candidate their citizens have selected f
or them, even if the voter lives in a state without a winner take all system."
```

student

```
train[train['generated']==0]['text'].iloc[10]
```

```
'Dear senator, Retain the Electoral College. The Electoral College consists of 538 electors and a majority of 270 electors
is is required to elect the President. Each state has hisher own electors which are chosen by the candidate political part
y. You should keep the Electoral College because you have certainty of outcome, and the President is everyones not just yo
urs.\n\nThe first reason why you should stay with the Electoral College is because you are certain that the outcome will b
e in favor of one of the candidates. A tie in the nationwide electoral vote may happen but it is very unlikely that it wil
l even though that 538number of electors in the Electoral College is a even numberS.3.For example in 2012\'s election, Oba
ma received 61.7 percent of the electoral votes compared to 51.3 percent of the popular cast for him and rodney because al
l states award electoral votes on a winnertakeall basis even a slight plurality in a state creates a landslide electoralvo
te victory in that stateS.3. However,because of the winnertakeall system in each state,candidates dont spend time in staes
they know they have no chance of winning, they only focus on the close,tight races in the "swing"statesS.2. But, the winni
ng candidates share of the Electoral College invariably exceeds his share of the popular vote.\n\nThe second reason you sh
ould keep the Electoral College is because the president is everyone\'s. The Electoral College requires a presidential can
didate to have transregional appeal. No region has enough electoral votes to elect a president by themselves. So for examp
le,a solid regional favorite,such as rodney was in the South,has no incentive to campaign heavily in those states for he g
ains no electoral votes by increasing his plurality in states he knows for sure that he will winS.3.A president with only
his regional apppeal is very unlikely to be a successful president. The residents of the other regions may feel like there
votes dont count or that he really isnt there president.\n\nIn conclusion, you should stay with the Electoral College simp
ly because you most likely not going to have a tie and because the president is everyone\'s.'
```

数据概览

外部数据：DAIGT V2 Train Dataset

组成成分：其中persuade_corpus是来自网络公开的学生写作议论文，而其余都是由不同LLM生成的文章。

persuade_corpus	25996
chat_gpt_moth	2421
llama2_chat	2421
mistral7binstruct_v2	2421
mistral7binstruct_v1	2421
original_moth	2421
train_essays	1378
llama_70b_v1	1172
falcon_180b_v1	1055
darragh_claude_v7	1000
darragh_claude_v6	1000
radek_500	500
NousResearch/Llama-2-7b-chat-hf	400
mistralai/Mistral-7B-Instruct-v0.1	400
cohere-command	350
palm-text-bison1	349
radekgpt4	200

Sources (please upvote the original datasets!):

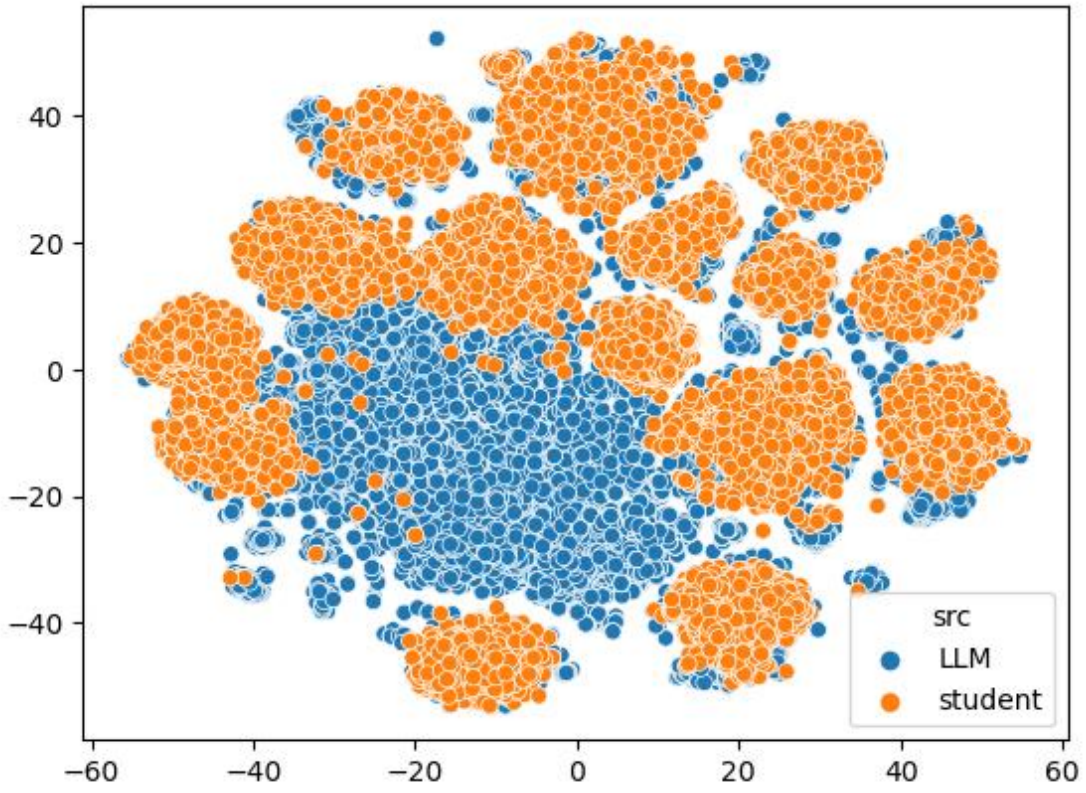
- Text generated with ChatGPT by MOTH (https://www.kaggle.com/datasets/alejopaullier/daigt-external-dataset)
- Persuade corpus contributed by Nicholas Broad (https://www.kaggle.com/datasets/nbroad/persaude-corpus-2/)
- Text generated with Llama-70b and Falcon180b by Nicholas Broad (https://www.kaggle.com/datasets/nbroad/daigt-data-llama-70b-and-falcon180b)
- Text generated with ChatGPT and GPT4 by Radek (https://www.kaggle.com/datasets/radek1/llm-generated-essays)
- 2000 Claude essays generated by @darraghdog (https://www.kaggle.com/datasets/darraghdog/hello-claude-1000-essays-from-anthropic)
- LLM-generated essay using PaLM from Google Gen-AI by @kingki19 (https://www.kaggle.com/datasets/kingki19/llm-generated-essay-using-palm-from-google-gen-ai)
- Official train essays
- Essays I generated with various LLMs

License: MIT for the data I generated. Check source datasets for the other sources mentioned above.

https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset

数据概览

数据嵌入可视化



t-SNE 降维平面图

- Student: 即来自 persuade_corpus 的数据集，分别由15个主题构成
- LLM：利用同样的主题生成文本数据，并且额外增加主题来生成多样性数据

算法详解：线性模型

自定义分词器（tokenizer）

Hello, this is a test for Roberta tokenizer.

encode

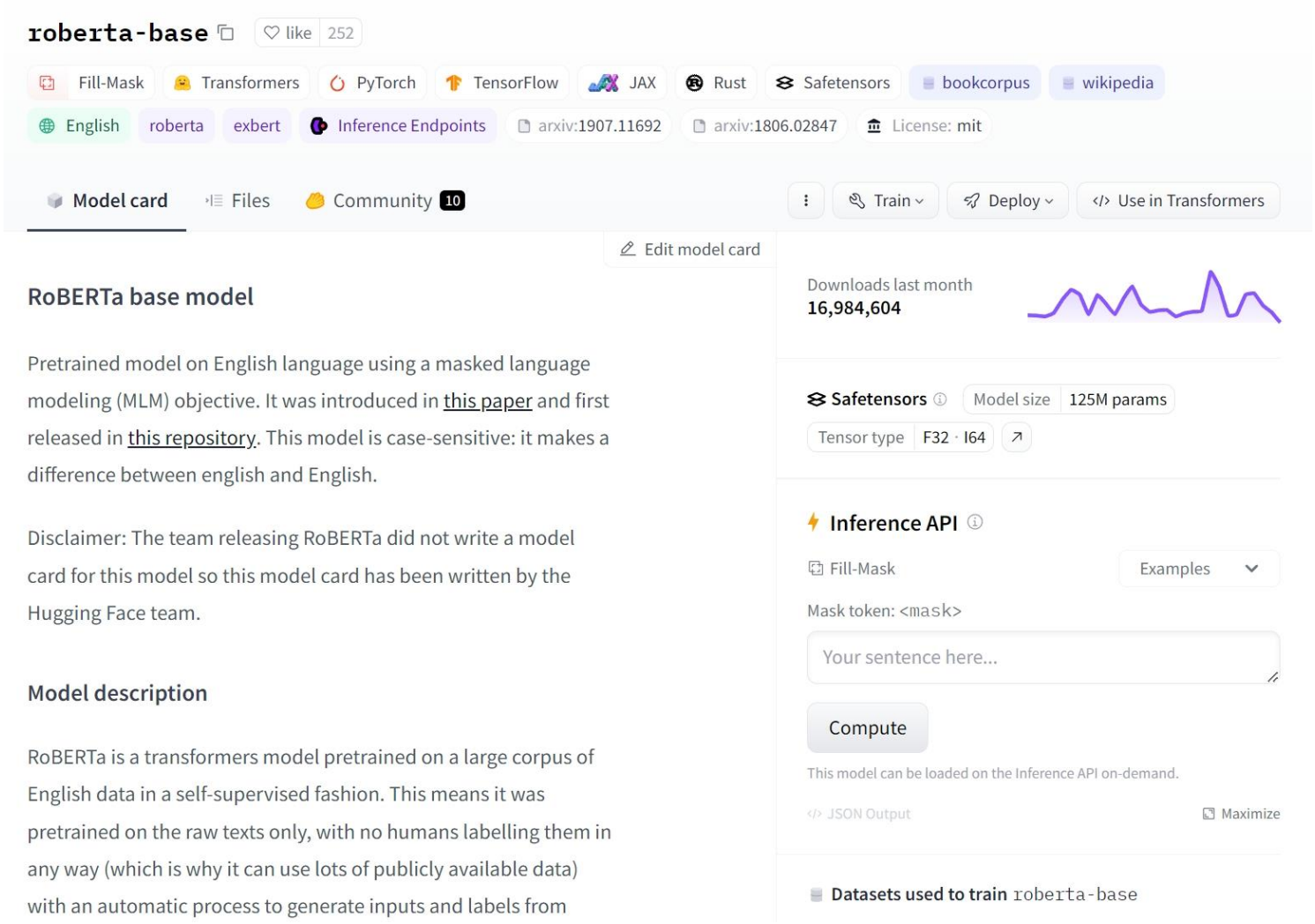
[0, 31414, 6, 42, 16, 10, 1296, 13, 1738, 102,

19233, 6315, 4, 2]

decode

<s>Hello, this is a test for Roberta tokenizer. </s>

https://huggingface.co/roberta-base/tree/main



算法详解：线性模型

BPETrainer

BPE（Byte Pair Encoding）算法用于词汇分割和子词（subword）单位的生成。BPE算法的核心思想是通过迭代地合并频率最高的字节对（或字符对）来减少整个数据集中的总字节（或字符）数量。在NLP中，这种方法特别适用于处理未知词（OOV，Out-Of-Vocabulary）问题，提高模型对罕见词或新词的泛化能力。

BPETrainer是用于训练BPE模型的工具，它实现了BPE算法的训练过程。以下是BPETrainer工作原理的简要概述：

- **初始化词汇表**：BPETrainer首先将训练数据集中的所有单词分解为基础字符（如字母或汉字字符）加上一个特殊的结束符号，然后统计并初始化一个基础词汇表，包括所有唯一的字符和它们的频率。
- **迭代合并**：BPETrainer接着迭代执行以下步骤：
  - 在当前词汇表的基础上，统计所有相邻字符对（bigram）的出现频率。
  - 找出出现频率最高的字符对，将这个字符对合并为一个新的符号（即这两个字符的组合），并将其添加到词汇表中。
  - 更新训练数据集中的单词，将选中的字符对替换为对应的新符号。
- **重复迭代**：重复步骤2的迭代过程，直到达到预设的词汇表大小或者合并次数。
- **生成模型**：完成所有迭代后，BPETrainer将基于最终的词汇表生成BPE模型。这个模型能够将新文本按照训练时学到的规则分割为子词单位。

```
# Adding special tokens and creating trainer instance
special_tokens = ["[UNK]", "[PAD]", "[CLS]", "[SEP]", "[MASK]"]
trainer = trainers.BpeTrainer(vocab_size=VOCAB_SIZE, special_tokens=special_tokens, )


# Creating huggingface dataset object
if VALID_MODE:
    dataset = Dataset.from_pandas(valid[['text']])
else:
    dataset = Dataset.from_pandas(test[['text']])

def train_corp_iter():
    """
    A generator function for iterating over a dataset in chunks.
    """
    for i in range(0, len(dataset), 1000):
        yield dataset[i : i + 1000]["text"]

# Training from iterator REMEMBER it's training on test set...
raw_tokenizer.train_from_iterator(train_corp_iter(), trainer=trainer)
raw_tokenizer.model.save('.')
```

算法详解：线性模型

N-Gram & TF-IDF

N-Gram是一种基于统计语言模型的算法。它的基本思想是将文本里面的内容按照字节进行大小为N的滑动窗口操作，形成了长度是N的字节片段序列。
每一个字节片段称为gram，对所有gram的出现频度进行统计，并且按照事先设定好的阈值进行过滤，形成关键gram列表，也就是这个文本的向量特征空间，列表中的每一种gram就是一个特征向量维度。常用的是二元的Bi-Gram和三元的Tri-Gram。

TF-IDF其实是两个词的组合，可以拆分为TF和IDF。
TF（Term Frequency，缩写为TF）也就是词频，即**一个词在文中出现的次数**，统计出来就是词频TF，显而易见，一个词在文章中出现很多次，那么这个词肯定有着很大的作用，但是文本中统计出来的TF 大都是：‘的’，‘是’这样的词，这些词可能会干扰统计分析，因此可以使用一些停用词的语料库等方法去掉这些无意义的词语。
在词频的基础上，要对每个词分配一个"重要性"权重。**最常见的词给予最小的权重，较常见的词给予较小的权重，较少见的词给予较大的权重。**这个权重叫做"逆文档频率"（Inverse Document Frequency，缩写为IDF），它的大小与一个词的常见程度成反比。
知道了"词频"（TF）和"逆文档频率"（IDF）以后，将这两个值相乘，就得到了一个词的TF-IDF值。某个词对文章的重要性越高，它的TF-IDF值就越大。

算法详解：线性模型

Top 10 3-grams only used by ai:

	3-gram	freq
0	sincerely your name	16549
1	writing to express	11987
2	today to express	6605
3	express my support	6365
4	judge richard a	5887
5	address city state	5267
6	city state zip	5173
7	support for abolishing	5092
8	a tour guide	5080
9	of our democratic	4469

Top 10 3-grams only used by human:

	3-gram	freq
0	snapped a picture	2947
1	main and his	2528
2	dr huang predicts	2484
3	team snapped a	2259
4	of anxious web	2079
5	anxious web surfers	2064
6	thousands of anxious	2057
7	flew over colonia	2040
8	dr paul beckman	1890
9	are the clouds	1808

```
# Getting vocab
vocab = vectorizer.vocabulary_
# Here we fit our vectorizer on train set but this time we use vocabulary from test fit.
vectorizer = TfidfVectorizer(ngram_range=(3, 5),
                             lowercase=False,
                             sublinear_tf=True,
                             vocabulary=vocab,
                             min_df = 2,
                            analyzer = 'word',
                            tokenizer = dummy,
                            preprocessor = dummy,
                            token_pattern = None, strip_accents='unicode'
                            )

tf_train = vectorizer.fit_transform(tokenized_texts_train_aug)
tf_test = vectorizer.transform(tokenized_texts_test_aug)
```

ngram_range : *tuple (min_n, max_n), default=(1, 1)*

The lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min_n <= n <= max_n will be used. For example an `ngram_range` of `(1, 1)` means only unigrams, `(1, 2)` means unigrams and bigrams, and `(2, 2)` means only bigrams. Only applies if `analyzer` is not callable.

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

算法详解：线性模型

线性模型：MultinomialNB & SGDClassifier

- 朴素贝叶斯：sklearn.naive_bayes.MultinomialNB
- Linear classifiers (**SVM**, logistic regression, etc.) with SGD training：sklearn.linear_model.SGDClassifier

```
bayes_model = MultinomialNB(alpha=0.023)
sgd_model = SGDClassifier(max_iter=35000, tol=1e-4, loss="modified_huber")

weights = [0.2,0.8,]

ensemble = VotingClassifier(estimators=[('mnb',bayes_model),
                                        ('sgd', sgd_model),
                                       ],
                            weights=weights, voting='soft', n_jobs=-1)
ensemble.fit(tf_train, y_train)
gc.collect()

final_preds = ensemble.predict_proba(tf_test)[:,1]
```

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

算法详解：线性模型

线性模型算法流程

Data: **DAIGT V2 Train Dataset** 经过相似度过滤（cosine > 0.92）

Tokenizer : 自定义BPE分词器训练，词汇表大小为5000

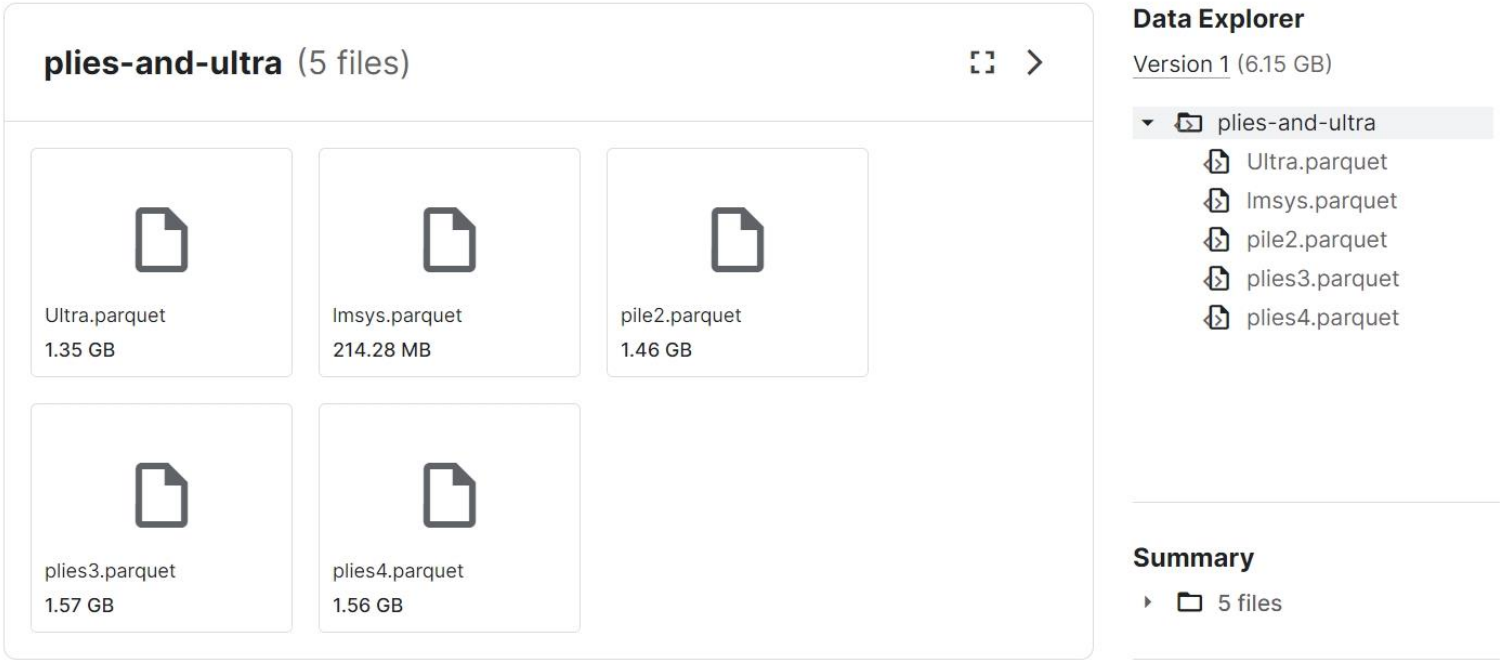
TFIDF : 采用ngram(3,5)的统计分布，最小词频为2

Linear model : MultinomialNB / SGDClassifier

算法详解：深度学习

外部数据：Pile and Ultra

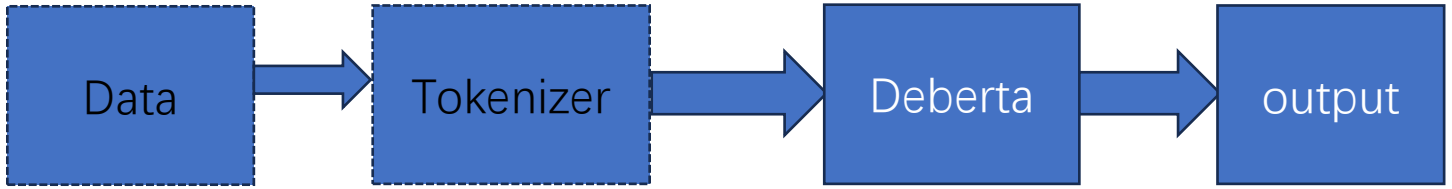
- Pile由许多不同的领域文本组成，包括书籍、github存储库、网页、聊天日志以及医学、物理、数学、计算机科学和哲学论文；
- Ultra取自清华大学多轮对话数据集，多个ChatGPT API进行相互对话，从而创造出更加自然、连贯的对话文本。



https://www.kaggle.com/datasets/canming/piles-and-ultra-data

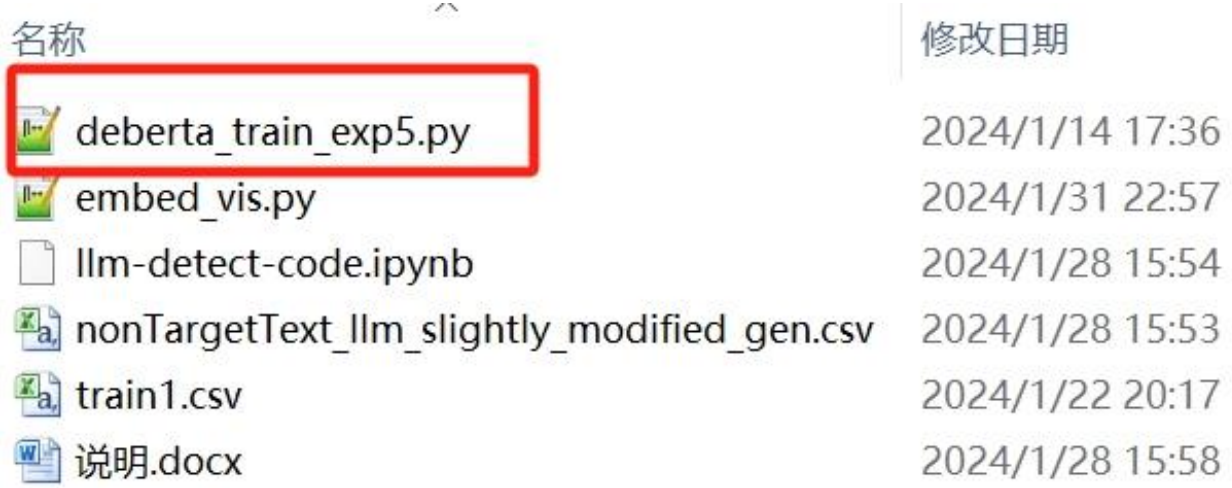
算法详解：深度学习

深度学习算法流程



Tokenizer : deberta-v3-small自带的分词器

Deberta : deberta-v3-small的二分类模型



算法详解：开源模型

DistilRoberta

In [51]:

```
metric_name = "roc_auc"
model_name = "distilroberta"#"deberta-large"
batch_size = 2

args = TrainingArguments(
    f"{model_name}-finetuned_v5",
    evaluation_strategy = "epoch",
    save_strategy = "epoch",
    learning_rate=2e-5,
    lr_scheduler_type = "cosine",

    optim="adamw_torch",
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    gradient_accumulation_steps=8,
    num_train_epochs=num_train_epochs,
    weight_decay=0.01,
    load_best_model_at_end=True,
    metric_for_best_model=metric_name,
    report_to='none',
    save_total_limit=2,

)
```

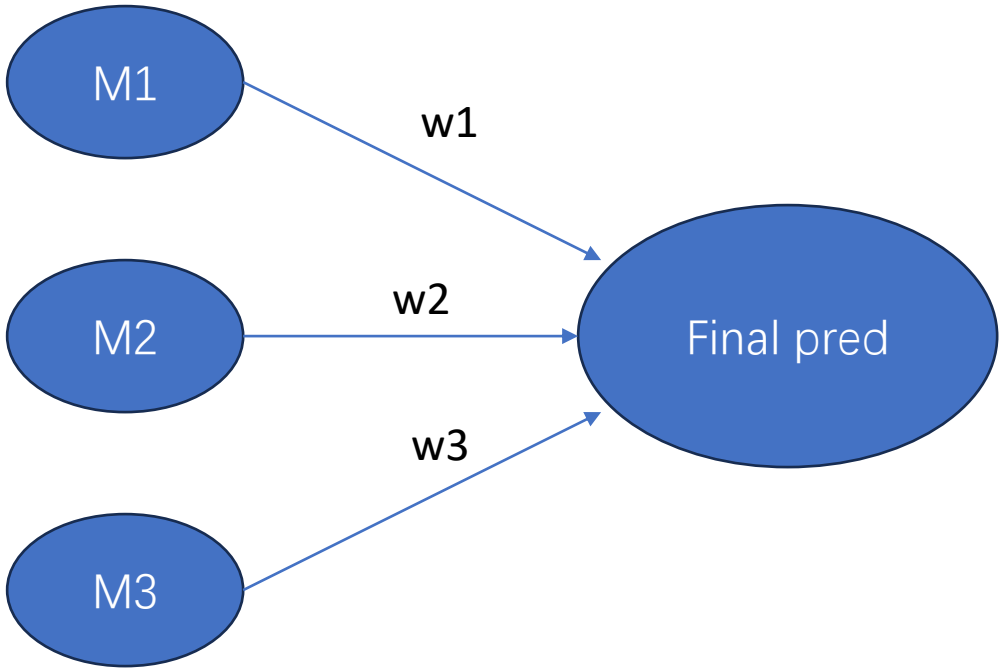
```
%%writefile distilroberta_infer.py#推理代码
import transformers
import datasets
import pandas as pd
import numpy as np
from datasets import Dataset
import os
from transformers import AutoModelForSequenceClassification, TrainingArguments, Trainer
import torch
from transformers import AutoTokenizer
##https://www.kaggle.com/datasets/mustafakeser4/detect-llm-models/versions/9
model_checkpoint = "/kaggle/input/detect-llm-models/distilroberta-finetuned_v5/checkpoint-13542"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
def preprocess_function(examples):
    return tokenizer(examples['text'], max_length = 512 , padding=True, truncation=True)
num_labels = 2
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=num_labels)
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    # Move your model and data to the GPU
model.to(device);
trainer = Trainer(
    model,
    tokenizer=tokenizer,
)
test = pd.read_csv('/kaggle/input/llm-detect-ai-generated-text/test_essays.csv')
test_ds = Dataset.from_pandas(test)
test_ds_enc = test_ds.map(preprocess_function, batched=True)
test_preds = trainer.predict(test_ds_enc)
logits = test_preds.predictions
probs = (np.exp(logits) / np.sum(np.exp(logits), axis=-1, keepdims=True))[:,0]
sub = pd.DataFrame()
sub['id'] = test['id']
sub['generated'] = probs
sub.to_csv('sub_nn.csv', index=False)
sub.head()
```

```
!python distilroberta_infer.py
```

https://www.kaggle.com/code/mustafakeser4/train-detectai-distilroberta-0-927

算法详解：集成

简单加权集成




```
p1['generated'] = p1['generated']*0.7+p3['generated']*0.2+p4['generated']*0.1
p1[['id', 'generated']].to_csv('submission.csv', index=False)
```