北京大学
PEKING UNIVERSITY

组会分享

张技轩

12.26

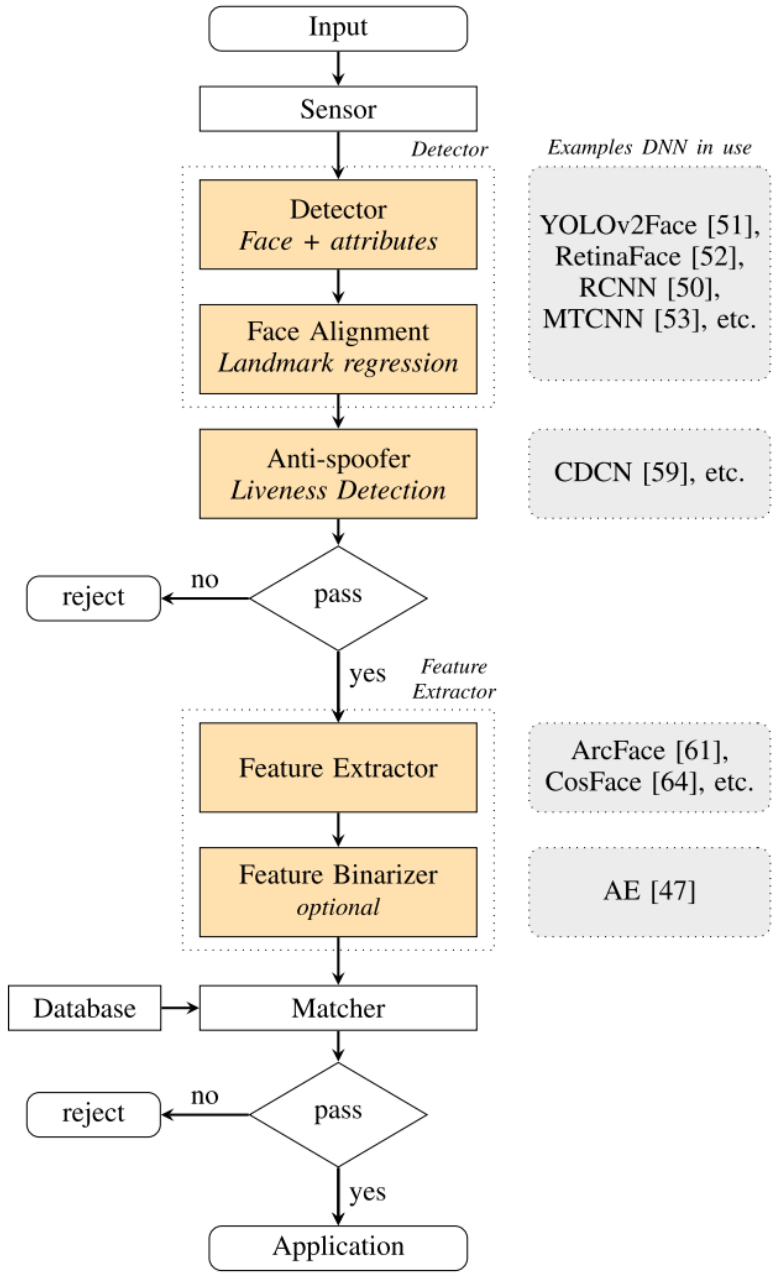
A Comprehensive Survey on BackdoorAttacks and Their Defenses in FaceRecognition Systems

[A Comprehensive Survey on Backdoor Attacks and Their Defenses in Face Recognition Systems (openreview.net)]

1.1 Face Recognition Systems(FRS)

FRS是一个多步骤的流程，它将输入（例如图像或视频流）处理到一个或多个生物识别模板中，用于下游任务，例如在合法数据库中进行人员认证或身份识别。

支持FRS的标准生物识别框架将生物识别流程分为三个不同的阶段：获取、提取和匹配。在此框架下，现代FRS由传感器、基于dnn的提取管道（本身由检测器、对齐步骤、反欺骗器、同名特征提取器以及可选的特征二值化器组成）和匹配器组成。

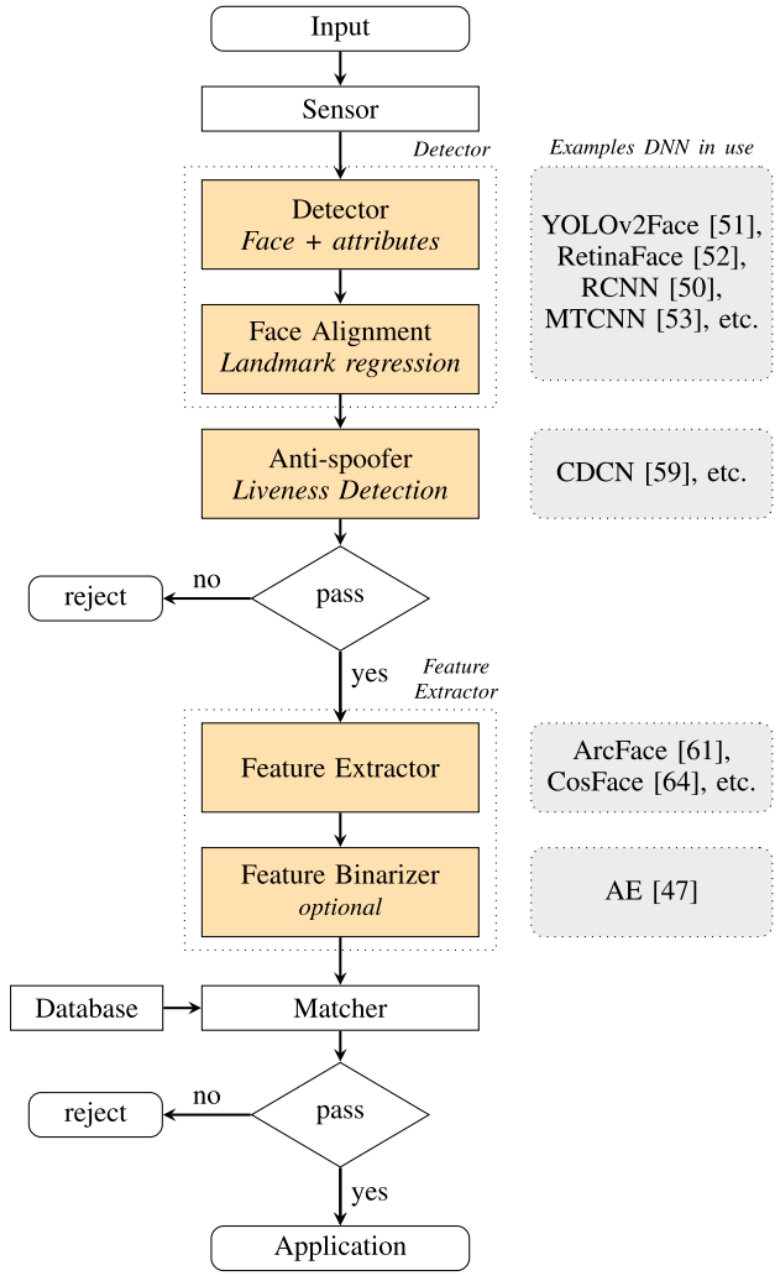


1.1 Face Recognition Systems(FRS)

传感器（如摄像头）获取图像并将其转发给人脸检测器。如果检测器在输入中识别出一个或多个人脸，则对这些人脸进行裁剪和对齐，以适应用于存储和操作二维和三维人脸数据的标准化表示。

之后，反欺骗器扫描预处理后的人脸以确定其真实性。此步骤旨在捕获意外输入和恶意输入。如果人脸被认为是真实的，则继续进行特征提取步骤。最后，匹配器处理这些向量，根据预先确定的阈值做出接受/拒绝的决定，并通知与FRS接口的下游任务。分数低于阈值表示匹配。

还有一个可选的特征二值化模块可以添加到特征提取器之后。二值化将实值特征向量转换为二进制格式，从而实现特定的安全特性，这个可能与类似与模糊承诺的加密方法有关。



1.2 现代FRS用到的DNN

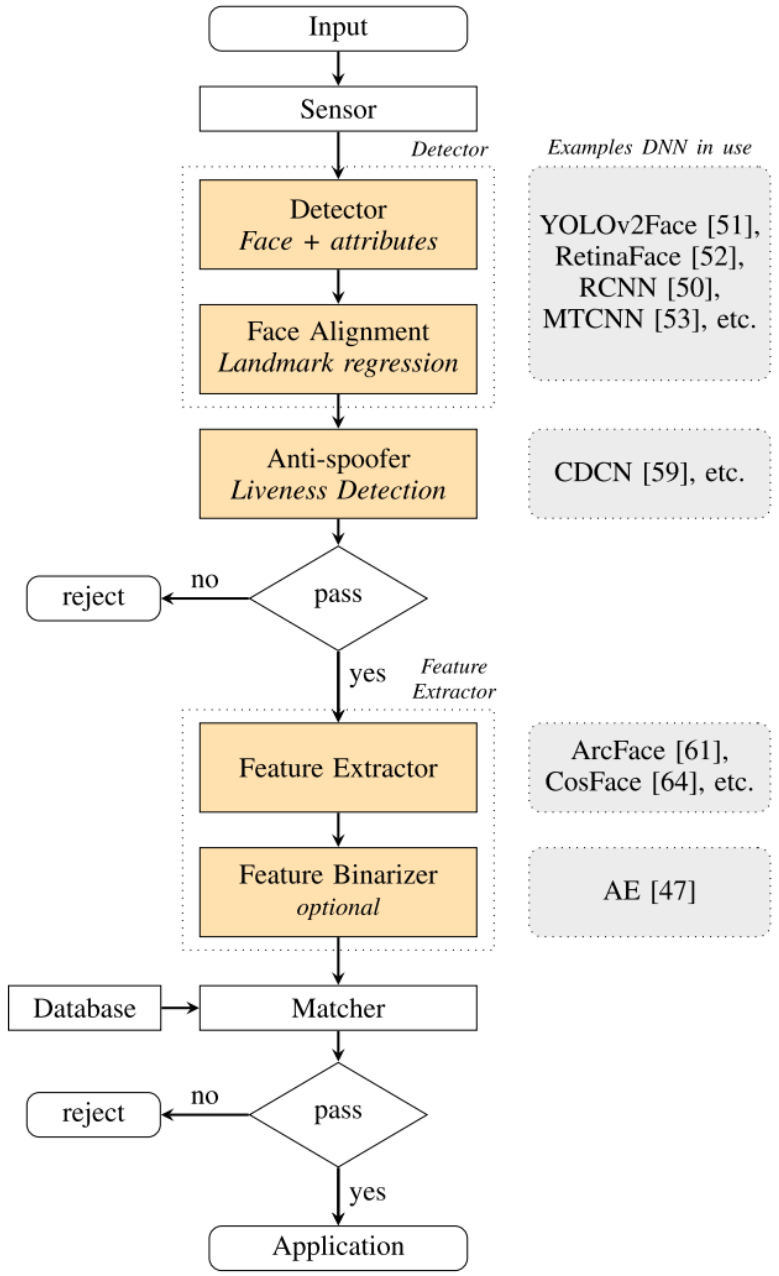
Detector通常依靠主干卷积神经网络（CNN）来识别和定位图像中的一个或多个人脸。DNN对边界框的坐标进行回归，并预测其相关的属性或标签。

Face Alignnment通常涉及CNN，预测（例如通过回归）人脸标记的坐标，例如鼻子，眼睛和嘴巴的坐标。这些坐标随后被用于使面部符合规范形状。

Anti-spoofer依赖各种各样的模型，比如CNN和transformer验证检测到的人脸是否符合活性标准。虽然反欺骗器可以用来丢弃意外检测，但它首先被设计用于检测恶意呈现攻击，攻击者试图通过欺骗给定的身份来获得访问下游任务的权限。

特征提取器通常是经过特殊训练的CNN，用于将人脸图像映射到较低维的实值向量表示。这些embeddings是通过各种方法学习的，如度量或角边缘学习等，这有助于区分多个身份。因此，特征提取器构成了FRS的核心。

Feature Binarizer是FRS中的一个可选模块，它将真实值输出映射为二进制表示，从而实现重要的安全功能。



2.1 FRS中的安全风险

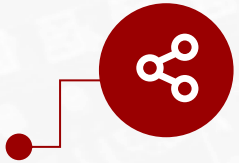
- **Integrity**

伪造生物特征

恶意代理试图通过管道传感器提交虚假的生物识别输入来欺骗FRS。例如，攻击者使用面罩、在纸上打印出的面部或三维结构来获得对基于身份验证的在线银行应用程序的未经授权访问。

重放攻击

重新提交生物识别。攻击者拥有FRS先前发送和接受的生物识别输入，绕过传感器，攻击者重新提交输入以错误地获得对FRS任务的访问权



劫持模型

恶意代理覆盖FRS中的一个或多个模型，例如，通过用恶意软件感染FRS，攻击者可能会强迫模型输出特定的特征表示，以混淆下游匹配器。

劫持特征

攻击者拦截并操纵由FRS模型生成的最终特征表示。例如，攻击者可以通过提取器或二值化器与匹配器之间的通信通道操纵信号。

2.1 FRS中的安全风险

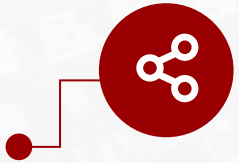
- **Integrity**

建立模板

攻击者以身份模板或模板库的存储位置为目标，分别用于FRS身份验证或识别。一旦模板被破坏，攻击者可能会覆盖FRS，匹配器将秘密地输出错误的决定

劫持模板

与劫持特征一样，攻击者在模板从存储位置传输到匹配器的过程中拦截并操纵模板。



劫持匹配器

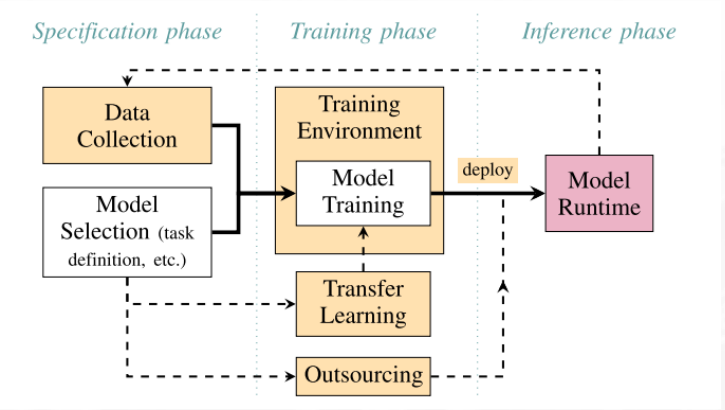
与模型劫持一样，攻击者可以操纵匹配器并改变其决策过程以产生期望的输出。

劫持决定

攻击者拦截并操纵匹配器的决策来劫持下游任务的功能

2.1 FRS中的安全风险

FRS中的安全风险风险既来自DNN的固有性质，也来自深度神经网络开发者在整个模型生命周期中倾向于依赖第三方的事实



对抗攻击

当攻击者与FRS中的部分DNN交互时，攻击者在数字空间或者是物理空间中，以一种无害或人眼难以察觉的方式操纵人脸输入，从而导致DNN产生错误的结果、表示或决策。对抗性攻击导致受害者DNN越过一些决策边界。

后门攻击

除了使用对抗性示例攻击FRS之外，攻击者还可以通过向FRS中发现的良性DNN注入隐形的错误行为来操纵它。一旦恶意DNN被毫无戒心的用户部署为FRS管道的一部分，攻击者就可以在推理过程中随时激活后门。例如，激活可以通过在传感器前持有一个触发补丁来实现。

2.2 FRS中的后门攻击

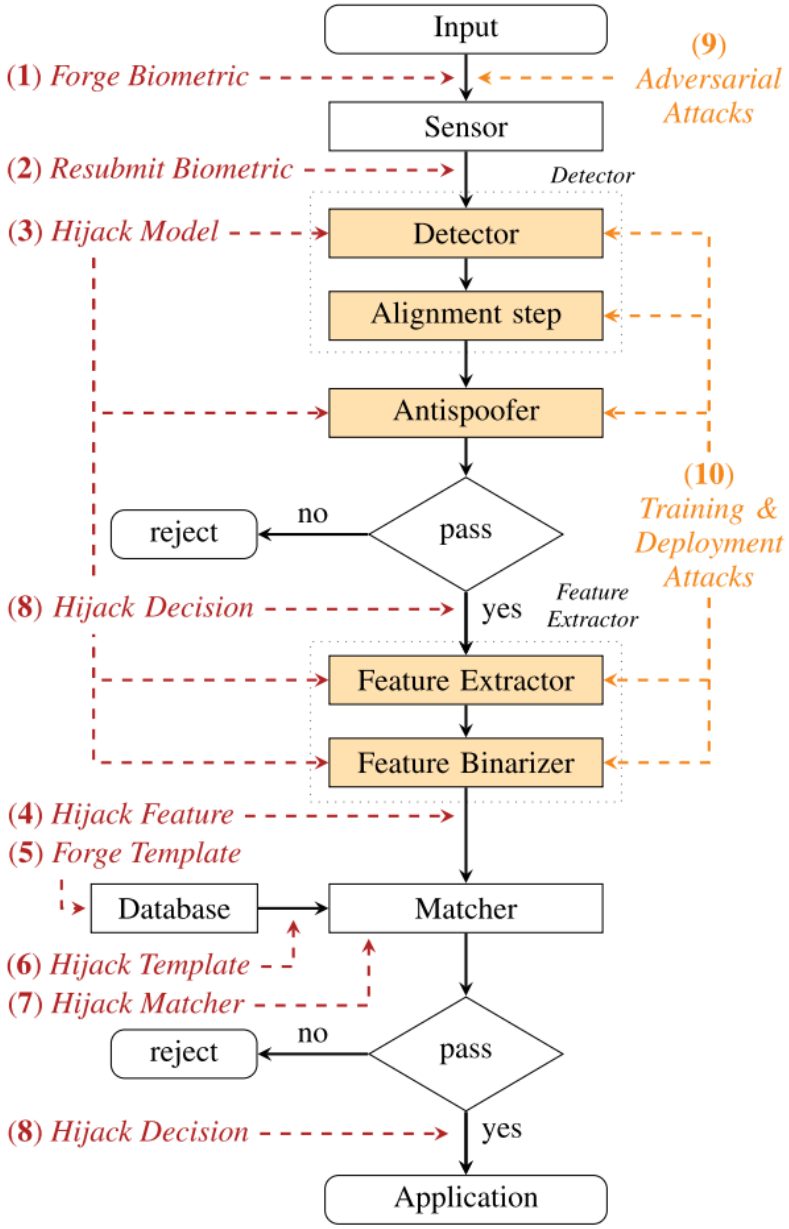
攻击者在FRS检测器中注入后门。攻击者在推理时操纵检测到的人脸数量，以逃避或欺骗检测器以及FRS中的所有后续步骤。（全局误分、局部误分、丢失目标、生成目标）

如果对齐模块与检测器分离，攻击者可能会在基于DNN的情况下进行后门对准，从而干扰面部地标的回归。这将导致一个不对齐的面，随后将无法通过FRS管道匹配。这将导致攻击者进行逃避攻击。

攻击者对反欺骗程序进行后门操作，使FRS在测试时错误地接受假面孔，这是典型的欺骗攻击。

攻击者在FRS的特征提取器DNN中注入一个后门来操纵下游匹配器的结果。这个后门改变了DNN的输出表示，使得它要么与攻击者的真实身份显著不同，从而逃避识别；要么与另一个人的真实身份接近，冒充受害者，以便进行错误的身份验证，并获得对其他机密或私人信息的访问权限。

对特征二进制器的后门攻击类似与特征提取器。



3.1 对FRS进行后门攻击

攻击途径

突出显示攻击者对受害者系统的假定知识和访问权限，这种情况发生在培训或部署阶段。此定义澄清了攻击者在何处执行攻击。

触发器特性

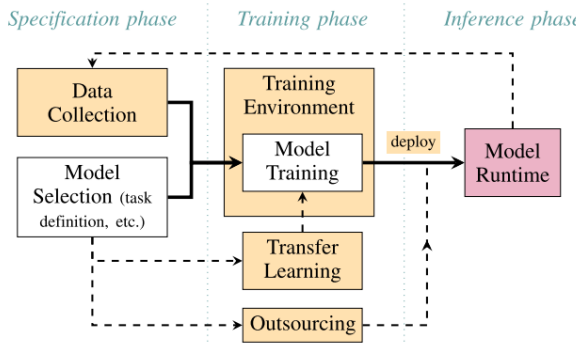
一旦受害者DNN被部署，攻击者选择何种输入损坏状态来激活后门。

插入方法

定义了攻击者如何操纵受害者DNN及其结束任务。例如，受害者可能已经将他们的DNN训练完全外包给恶意代理，尽管有广泛的访问权限，恶意代理只是操纵DNN的训练数据来注入后门。

3.1.1 后门攻击的攻击途径

- 后门攻击通常通过几个渠道之一发生，这些渠道包括提供给攻击者的知识和访问权限。



在这种情况下，攻击者缺乏对受害者DNN体系结构、训练环境或测试时间输入的控制甚至先验知识。因此，攻击者可能只通过在一个称为数据中毒的过程中隐藏易受攻击数据集的一部分中的模式来操纵受害者的训练数据

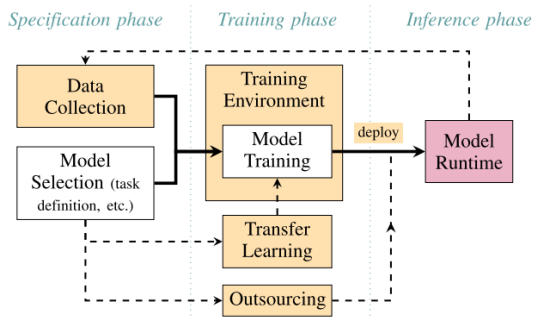
无法访问受害者训练数据的攻击者可能会转而攻击受害者DNN的训练环境。例如，攻击者可以在训练过程中操纵DNN的权重更新过程。因此，攻击者能够篡改模型并向其注入后门行为。在这种情况下，攻击者破坏训练环境以在延迟的时间内操纵DNN，通常被称为“盲目”后门攻击

模型外包是前两种渠道的结合，是MLaaS经济和整个外包的象征。在这种情况下，公司、机构或个人从第三方获得已经训练好的模型，而不是在内部开发。因此，模型外包的框架为恶意代理在DNN中嵌入后门提供了最大的自由

迁移学习是模型外包渠道的一个子类。一些最初为后一种情况设计的后门攻击也考虑了后门DNN被一些未知用户在称为迁移学习的过程中用于类似任务的场景：后门DNN被重用并通常（使用干净的数据）进行再训练，例如微调，架构修改和知识蒸馏方法。

攻击者可能在训练期间无法访问受害者的DNN、其训练数据集和其环境。然而，如果攻击者能够访问经过训练的模型的部署管道或易受攻击的机器上的存储位置，那么攻击者仍然可以将后门注入到该模型中。这种攻击被归类为部署攻击，攻击者在运行时操纵DNN的训练权重或其结构，对所需的后门进行编码。

3.1.2 插入方法



TRAINING ENVIRONMENT MANIPULATION

除了在训练时注入带有数据中毒的后门以后门DNN外，攻击者还可能篡改DNN的训练环境本身。在这种情况下，应用于FRS DNN的后门注入方法操纵以下一个或多个过程：(1)数据增强策略；(2)损失函数计算；(3)梯度反向传播

DATA POISONING

攻击者希望在DNN中注入后门，但无法直接访问它，可能依赖于操纵DNN的训练数据集来实现。这种技术被称为“数据中毒”，涉及对训练数据点的操纵，使它们包含一个或多个触发模式，以供受害者DNN学习。(1)有标签中毒;(2)无标签中毒

MODEL ARCHITECTURE AND WEIGHT MANIPULATIONS

当数据和训练环境都不可访问时，攻击者可能会求助于操纵目标DNN的权重或架构。后门注入通常发生在目标DNN的部署期间（例如，当它被存储或传输到推理环境时）。(1)模型架构操作；(2)目标权重扰动。

3.1.2 DATA POISONING

两者之间的区别取决于攻击者是否也通过修改数据点及其标签来改变数据集的一部分，或者仅修改所述数据点。

有标签中毒

攻击者从源、良性类中选择数据点，用触发模式操纵它们，然后将它们的源标签修改为给定的目标标签，具有更高的攻击成功率

无标签中毒

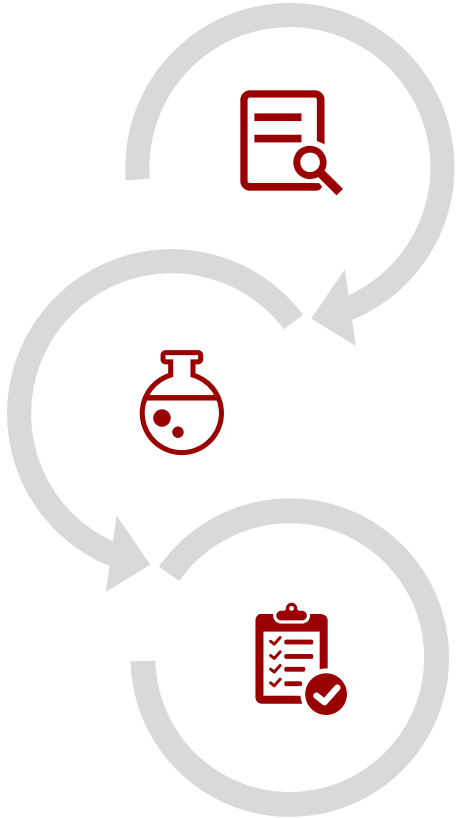
攻击者直接从目标类中选择数据点，并按照模式修改它们，具有更高的后门攻击的隐蔽性

3.1.2 TRAINING ENVIRONMENT MANIPULATION

北京大学 PEKING UNIVERSITY

数据增强策略

操纵深度神经网络的数据增强策略是一种基于训练环境的注入方法，最接近于数据中毒。如果攻击者无法控制受害者的数据收集，他们可能会转而攻击训练环境的数据加载器。



损失函数计算

攻击者设计了一个恶意的损失函数，并注入它来代替受害者的原始损失。攻击者的损失通常会在目标DNN的学习过程中引入一个额外的恶意目标。

梯度反向传播

直接修改受害者DNN的梯度反向传播步骤.攻击者识别源和目标输入身份的显著特征，并迫使受害者DNN最小化两者之间的距离，从而导致恶意特征冲突。因此，源输入变成了后门

训练管道后门可以实现更强大、更隐蔽的攻击，特别是当攻击者通过模型外包通道发起攻击时，会发生数据中毒。例如，在FR任务中，复合攻击利用恶意数据增强策略和毒物标签中毒策略的组合来增强其后门的隐身性。

模型架构操作

模型架构操作通常涉及将恶意子网嫁接到其他良性DNN上。TrojanNet攻击通过训练检测输入人脸图像中的后门触发器的子网，在FRS背景下演示了这种方法。只要检测到触发器，子网输出层的激活就会压倒受害FRS的激活。

目标权重扰动

攻击者可以在受害者DNN中识别特定的神经元集，而不是从单独的后门子网开始。这种注射方法被称为目标权重扰动攻击。这些方法的选择过程通常识别由所谓的非活动或休眠神经元组成的DNN中的冗余子网或路径。一旦识别出来，这些DNN元素就会被放大，在攻击者的后门触发器存在的情况下激活，通过受害者DNN创建恶意捷径。

3.1.3 触发器类型

格式

patch：在输入中插入后门

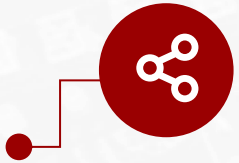
blended：混合触发器不会在某些位置替换内容，而是在目标图像的整个范围内叠加漫射模式。

warping：通过移动目标图像的像素，将图像变形作为后门触发器执行

插入空间

pixel、frequency：在数字空间中，触发器可以包含对图像像素的直接操作，如由传感器捕获的，或者干预图像的频率空间。

physical：物理攻击通常旨在将数字设计的触发器移植到物理空间中



可视性

为了追求更高的隐蔽性，使后门攻击中的patch人眼不可见，比如增加patch的透明度

设计

handcraft：过去的触发器基本都是手工的，即任务、数据集和目标模型不会影响触发器的构造。

optimized：使用优化技术设计更复杂的触发器

3.1.3 触发器类型

补丁位置

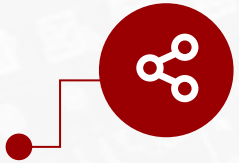
static：后门输入中触发器的位置是DNN中后门激活的必要条件之一，后门攻击的限制性工作条件

dynamic：通过混淆旨在重建后门触发器的防御措施来增强后门的隐蔽性。

语义

non-semantic：独立于底层任务和数据集的触发器的攻击

semantic：语义后门攻击利用训练数据集的上下文来创建相关触发器，使它们更不易被发现，尤其是人眼



目标

all-to-one：后门的目标独立于源图像

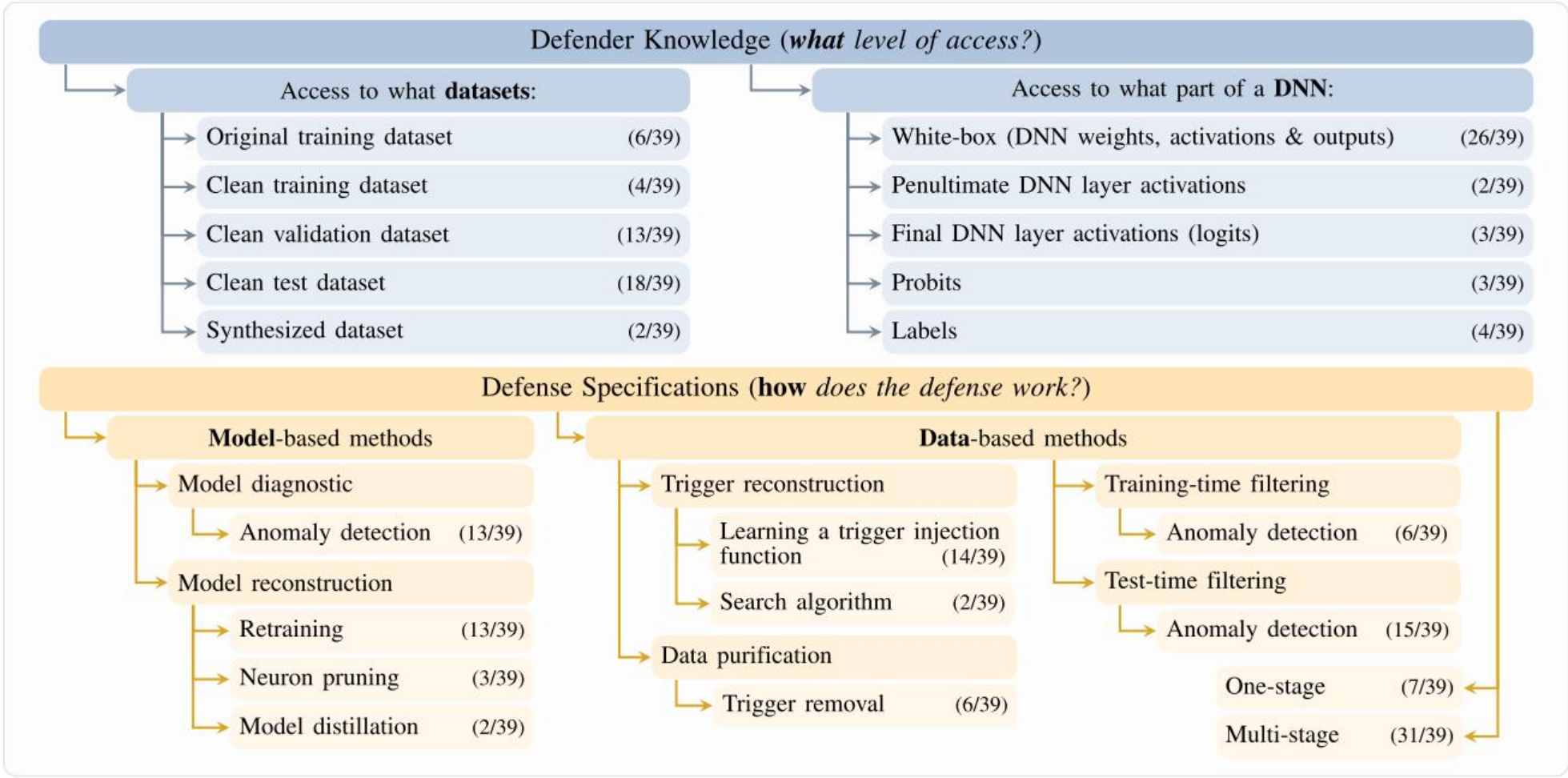
all-to-all：多对多后门使用独特的触发模式来产生不同的预测，这取决于被后门输入的源类（或FRS上下文中的身份）

特异性

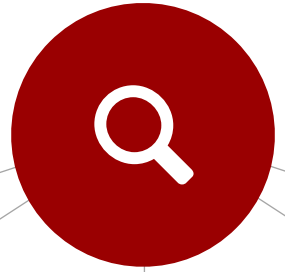
sample-agnostic：与样本无关而与触发器模式有关，

sample-specific：学习后门注入函数，根据底层输入定制后门模式

4.1 在FRS中的后门防御



4.1.1 ACCESS TO DATA



原始训练数据	干净训练数据	干净验证数据	干净测试数据	综合数据
这种设置中的防御通常是为了防止数据中毒漏洞，在将模型训练外包给第三方的情况下，知识产权可能会禁止DNN防御者访问提供商的专有数据，情况比较少。	通常是在DNN防御者也向第三方DNN训练者提供训练数据集的情况下假设的。因此，防御者可以通过询问训练者的DNN来验证训练者的DNN。	防御者从训练集中拿出一些干净的数据用于验证目的，以验证第三方的DNN训练。	探索使用干净的测试数据（即来自与用于训练DNN的数据集分布不同的数据集）作为防御后门的基础。当潜在任务相似时，在防御设计中使用训练、验证和测试数据之间只有很小的实际差异。	这种设置源于与数据收集相关的后门漏洞。相反，这些防御避开了数据收集过程，而是通过模型反演或蒸馏直接从仔细检查的DNN中合成数据点。

4.1.2 ACCESS TO A FRS DNN

白盒防御

DNN防御者可能需要完全访问模型的权重参数、激活和输出。也就是说，DNN的内部状态暴露在防御者的审查之下（例如，在训练或测试时间输入的前向传播期间）

倒数第二层激活

一些防御仅依赖于部分白盒访问，比如倒数第二层

DNN的logit、probit或label输出

在模型外包和托管的背景下，访问DNN的logits或probits可能是一个不切实际的假设

4.2.2 FRS BACKDOOR DEFENSES

北京大学 PEKING UNIVERSITY

基于模型的防御

模型诊断防御旨在检测模型是否包含后门。这些防御基于特定设计的度量来执行假设检验，该度量有助于区分良性和后门DNN的行为。
模型重建防御提供了从功能齐全的DNN中移除后门的方法。即使没有后门诊断，DNN防御者仍可能继续进行重建，希望消除DNN中潜在的后门。(1)模型再训练，(2)神经元修剪，(3)模型蒸馏

基于数据的防御

触发器重建围绕着防御者询问带有精心制作的输入的模型，以揭示和合成后门触发器。这种类型的防御是关键步骤，通常与其他后门防御共同使用。(1)学习触发器注入函数和(2)搜索算法。
如果防御者不能使用基于模型的防御或在输入级别执行后门检测，他可能会尝试不加选择地清除用户发送给被防御的DNN的输入。在实践中，防御者会更改传入输入的内容，使原始语义保持不变，同时销毁其中包含的触发器

4.2.2 FRS BACKDOOR DEFENSES

北京大学 PEKING UNIVERSITY

训练阶段过滤

每当训练一个DNN容易受到数据中毒的影响时，一种防御方法是从训练数据集中过滤异常值。因此，负责用不可信数据集训练DNN的DNN防御者可以通过观察它们与学习的DNN的相互作用来去除可疑的训练数据点。

测试阶段过滤

DNN防御者必须防御潜在的后门DNN（例如，在外包其培训之后），可能会过滤可能恶意用户发送的测试阶段的输入。测试阶段过滤与数据净化的不同之处在于，防御者假设有能力区分有害和良性输入，这是数据净化方法中不存在的假设。

1.防御的适用性有限
2.主要集中在：基于补丁的、样本无关的、多对一的和非语义的后门攻击
3.大多数都是白盒攻击

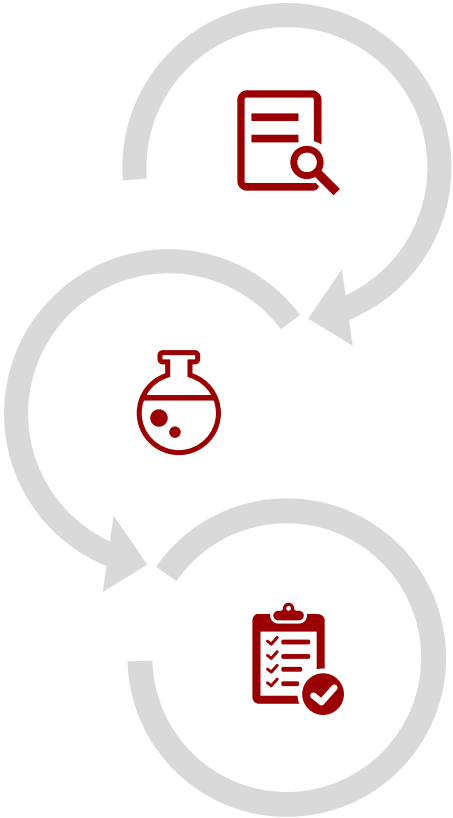
生成人脸数据集及其问题

由于隐私或伦理问题、标签问题或数据集偏差，人工人脸识别数据集的生成近年来变得越来越重要。但是，生成式DNN不能为DNN用户提供针对后门的保护，生成模型既容易受到后门攻击，也容易成为数据中毒攻击的载体。这样的模型可以被训练成为受害者生成后门数据的恶意生成模型。

新的后门攻击防御方法

除了后门攻击人脸识别任务的背景之外，最近的研究还探索了基于不可感知性、样本特异性和稀疏性设计更强的触发器，有时在图像分类中常见的良性过程可以打败现有的防御。

5.2 FRS中后门攻击文献的局限性



人脸识别只是FRS的一部分

绝大多数研究将人脸提取器dnn作为孤立的模型，很少研究反欺骗器。先前的后门攻击和防御没有考虑连续的任务集（人脸检测，提取和对齐，反欺骗，特征提取，以及可选的二值化）以及与每个任务相关的后门风险

人脸分类并不是现实

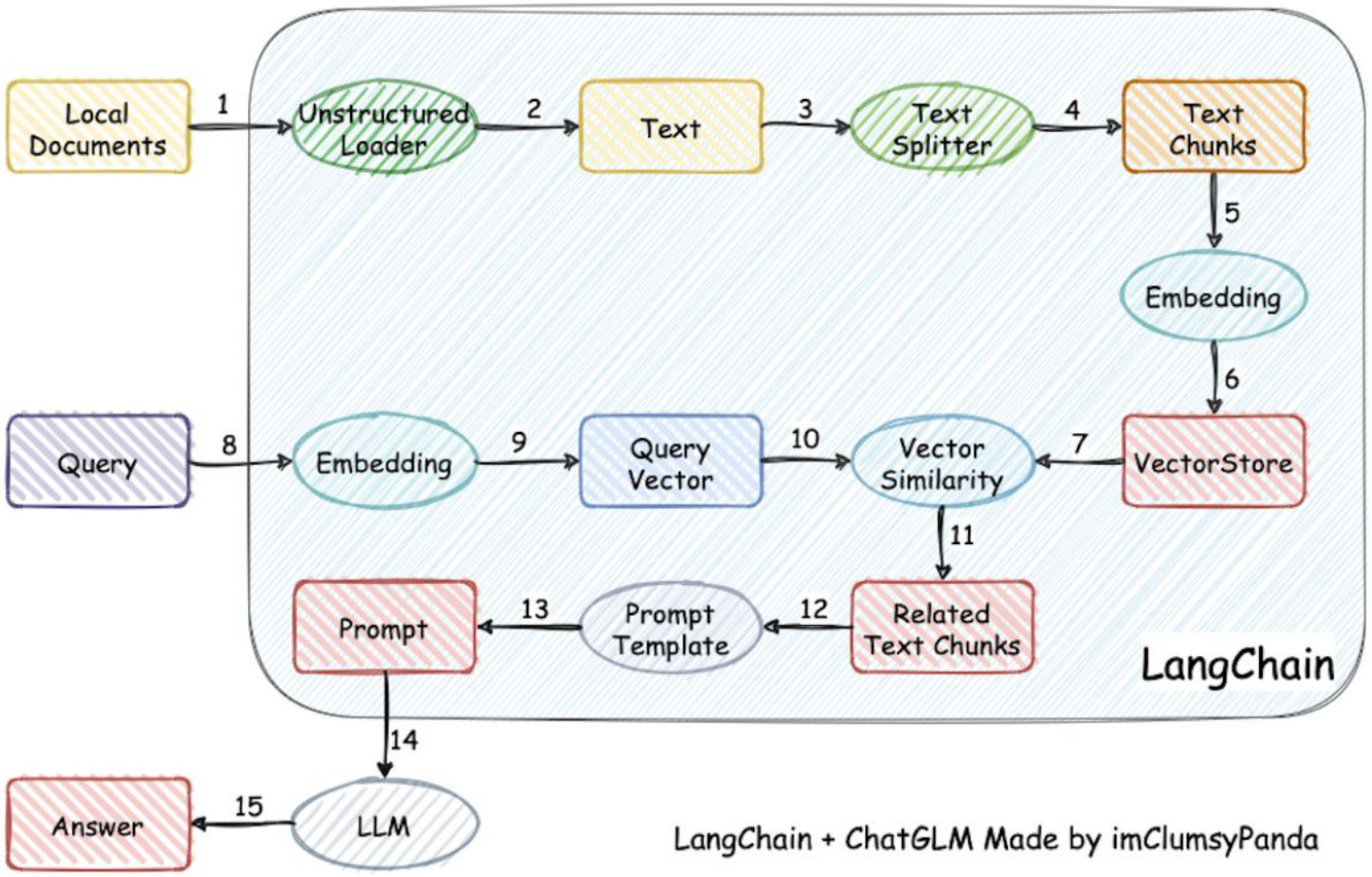
(1)现代人脸识别不是一个分类任务，
(2)人脸识别是一个开放集问题。

后门防御可能会改变后门安全的方向

后门防御的过程可能会员用到其它的DNN模型，但其它的DNN模型或者生成模型也无法保证安全。

Langchain+ChatGLM

CHATCHAT PIPELINE



文档解析与拆分

北京大学 PEKING UNIVERSITY

文档拆分主要用到ChineseRecursiveTextSplitter.py函数继承自Langchain 的RecursiveCharacterTextSplitter。其主要功能是通过使用正则表达式，基于不同的分隔符（例如标点符号、空格等）来分割中文文本。这个分割器主要用于处理长文本，将其分割成更小的块，并且可以根据用户需求选择是否保留分隔符。

1.使用正则表达式来分割文本，并提供选项决定是否保留分隔符

```python
def _split_text_with_regex_from_end(
    text: str, separator: str, keep_separator: bool
) -> List[str]:
    # Now that we have the separator, split the text
    if separator:
        if keep_separator:
            # The parentheses in the pattern keep the delimiters in the result.
            _splits = re.split(f"({separator})", text)
            splits = ["".join(i) for i in zip(_splits[0::2], _splits[1::2])]
            if len(_splits) % 2 == 1:
                splits += _splits[-1:]
            # splits = [_splits[0]] + splits
        else:
            splits = re.split(separator, text)
    else:
        splits = list(text)
    return [s for s in splits if s != ""]
```

文档解析与拆分

北京大学 PEKING UNIVERSITY

2.处理多个分隔符，按照从细到粗的顺序分割文本

```python
class ChineseRecursiveTextSplitter(RecursiveCharacterTextSplitter):
    def __init__(
        self,
        separators: Optional[List[str]] = None,
        keep_separator: bool = True,
        is_separator_regex: bool = True,
        **kwargs: Any,
    ) -> None:
        """Create a new TextSplitter."""
        super().__init__(keep_separator=keep_separator, **kwargs)
        self._separators = separators or [
            "\n\n",
            "\n",
            "。|！|？",
            "\.\s|\!\s|\?\s",
            "；|;\s",
            "，|,\s",
        ]
        self._is_separator_regex = is_separator_regex
```

文档解析与拆分

北京大学 PEKING UNIVERSITY

3.递归分割长文本

```
_good_splits = []
_separator = "" if self._keep_separator else separator
for s in splits:
    if self._length_function(s) < self._chunk_size:
        _good_splits.append(s)
    else:
        if _good_splits:
            merged_text = self._merge_splits(_good_splits, _separator)
            final_chunks.extend(merged_text)
            _good_splits = []
        if not new_separators:
            final_chunks.append(s)
        else:
            other_info = self._split_text(s, new_separators)
            final_chunks.extend(other_info)
```

文档解析与拆分

4.处理文本的重叠和合并

```
if _good_splits:
    merged_text = self._merge_splits(_good_splits, _separator)
    final_chunks.extend(merged_text)
```

针对这一模块，还有其它分词器的实现，例如AliTextSplitter，该函数实现依然继承了langchain的CharacterTextSplitter函数，并且采取的文档语义分割模型为达摩院开源的nlp_bert_document-segmentation_chinese-base，属于nlp语义分割，需要下载modelscope库来实现。

另外还有chinese_text_splitter.py和zh_title_enhance.py的实现.

其它流程

文本向量化

项目使用了xinference来安装调用了包括bge-large-zh-v1.5在内的很多embedding模型来进行文档向量化。通过在model_setting.yaml中修改默认选用的大模型以及Embedding，然后在xinference中下载相应的模型，即可在chatchat中实现调用。
另外，通过调研发现，FlagEmbeddin库也实现了比较多的文本向量化模型和文本检索模型。

语义检索

chatchat中设定了知识库匹配向量数量VECTOR_SEARCH_TOP_K为3，且知识库匹配相关度阈值SCORE_THRESHOLD为2。
Vectorstore：相似性得分阈值检索
Ensemble：结合BM25和Vectorstore

北京大学
PEKING UNIVERSITY

谢谢大家

张技轩

2024.12.26