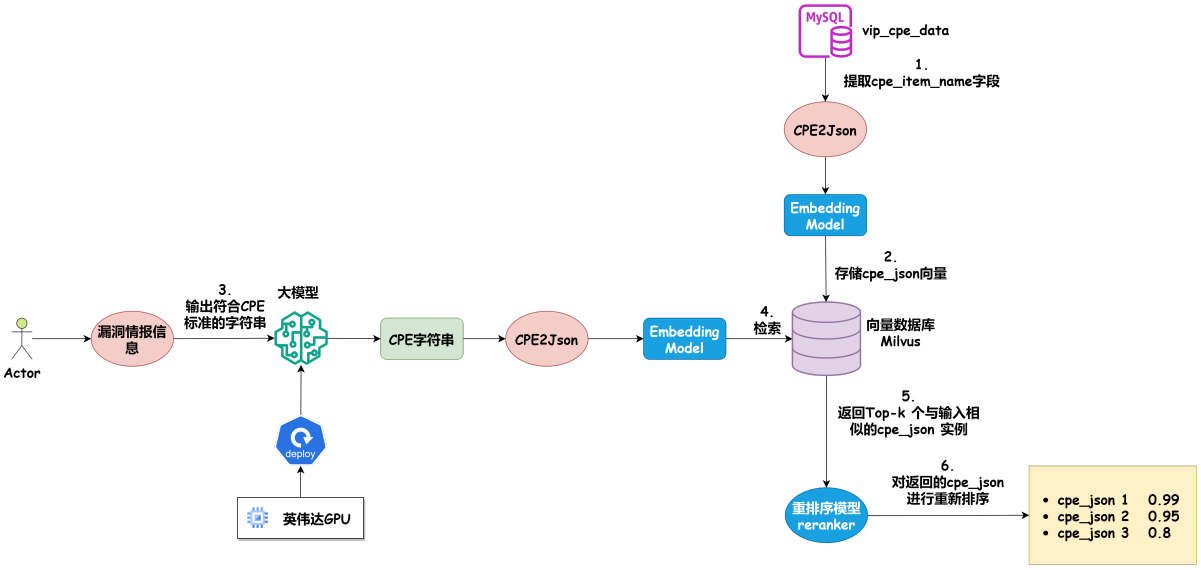
一、总流程结构



整个流程可以分为三个主要步骤：**数据存储**、**大模型提取与格式化**、**查询与匹配**。具体步骤如下：

---

1. 数据存储流程

步骤概述：将CPE数据从MySQL数据库中提取，转化为适合向量化的格式，存入向量数据库，以便后续的相似性检索。

流程详细：

1.1. **数据提取**：从MySQL数据库表`vip_cpe`中提取字段`cpe_item_name`（如`"cpe:/a:jenkins:fortify_on_demand:1.0::~~~jenkins~~."`）。

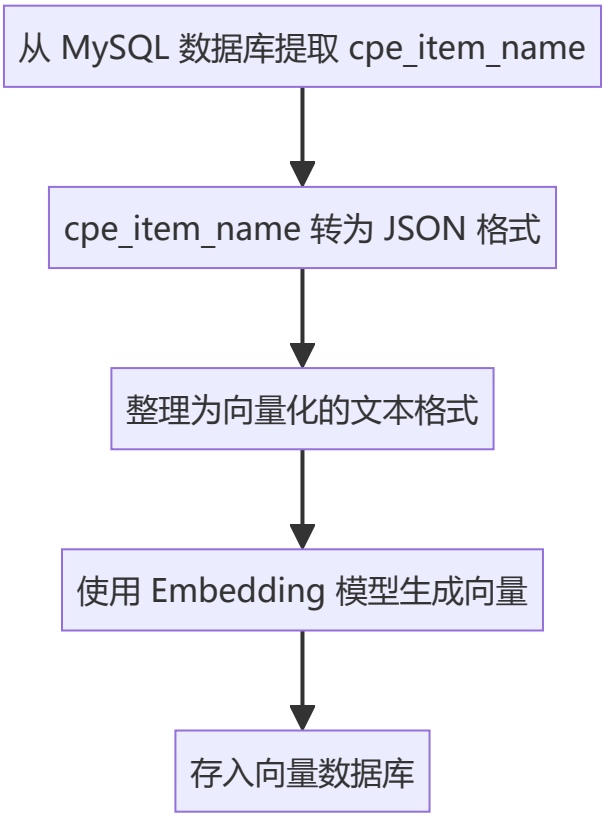
1.2. **JSON格式转换**：使用`cpe2json`方法将`cpe_item_name`字段转化为JSON格式的字符串，确保结构化的字段表示。

```
{
    "part": "a",
    "vendor": "jenkins",
    "product": "fortify_on_demand",
    "version": "1.0",
    "update": "",
    "edition": "~~~jenkins~~.",
    "language": ""
}
```

1.3. **向量化处理**：将JSON格式的CPE数据转换为适合向量化的文本格式，并输入Embedding模型生成向量表示。

1.4. **向量存储**：将生成的向量及原始数据存入向量数据库，以便后续相似性检索。

流程图：



---

2. 大模型CPE提取与格式化流程

步骤概述：使用大模型从漏洞情报文本中自动提取符合CPE格式的字符串，并将其格式化为JSON对象，便于后续查询。

流程详细：

2.1. **漏洞情报收集**：从外部情报源或内部系统中获取漏洞描述文本。

2.2. **大模型解析**：使用大模型解析漏洞情报文本，识别并提取所有符合CPE格式的字符串。

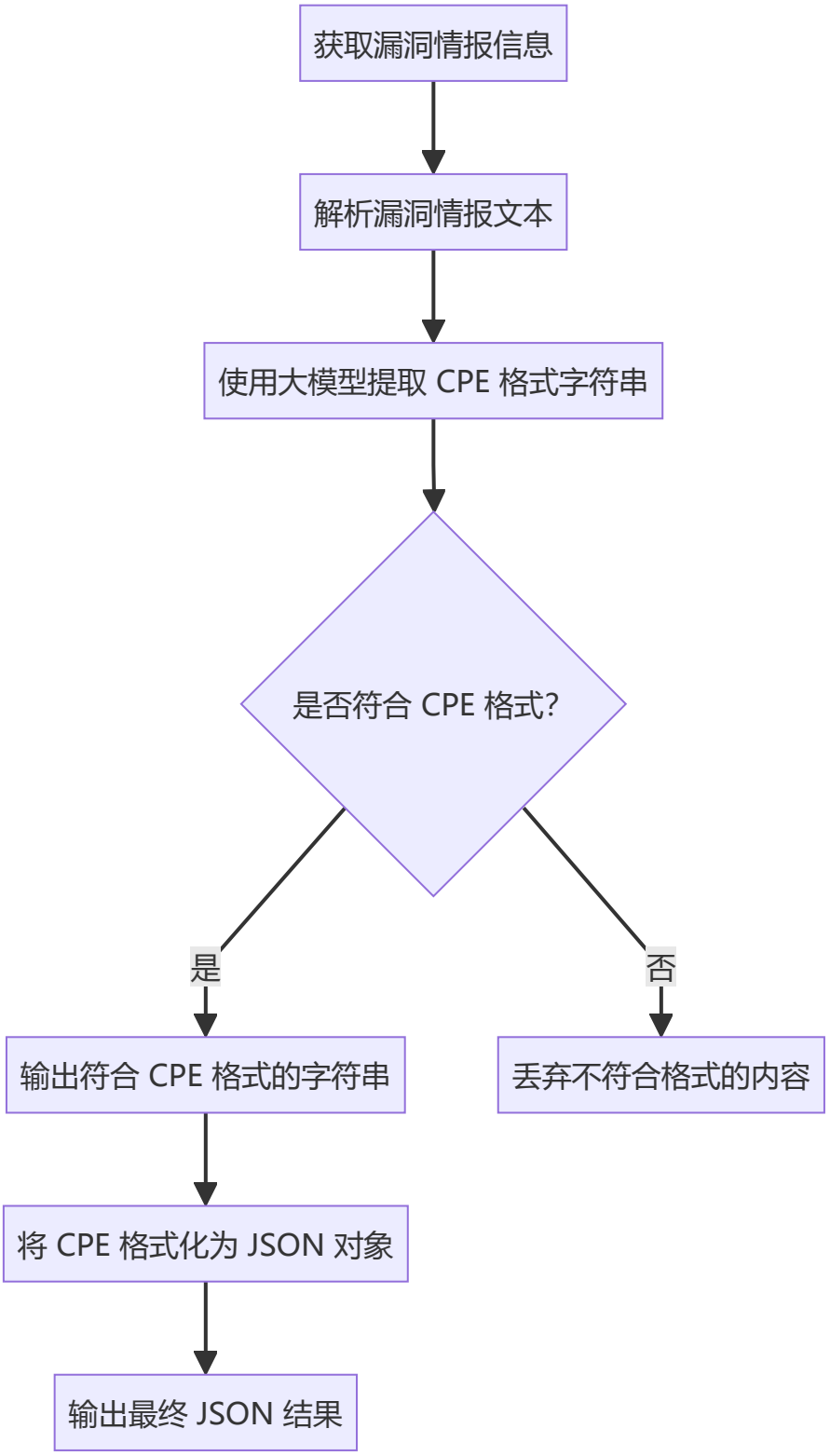
2.3. **格式校验**：判断提取的内容是否符合`cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}`标准格式。

2.4. **JSON格式化输出**：将符合CPE格式的字符串转换为JSON对象，并以`cpe`键为单一键输出JSON结果。

提取要求：

- **输出格式**：结果为JSON对象，包含单一键`cpe`，其值为包含提取CPE字符串的数组。
- **字段严格性**：若字段不完整，则保持原样，不进行补充或修改。

流程图：



---

3. 查询与匹配流程

步骤概述：在接收到输入的CPE字符串后，转化为向量表示，并在向量数据库中进行相似性检索，以返回最匹配的CPE项。

流程详细：

3.1. **接收输入CPE**：接收由大模型从漏洞情报中提取的CPE字符串或其他来源的CPE。

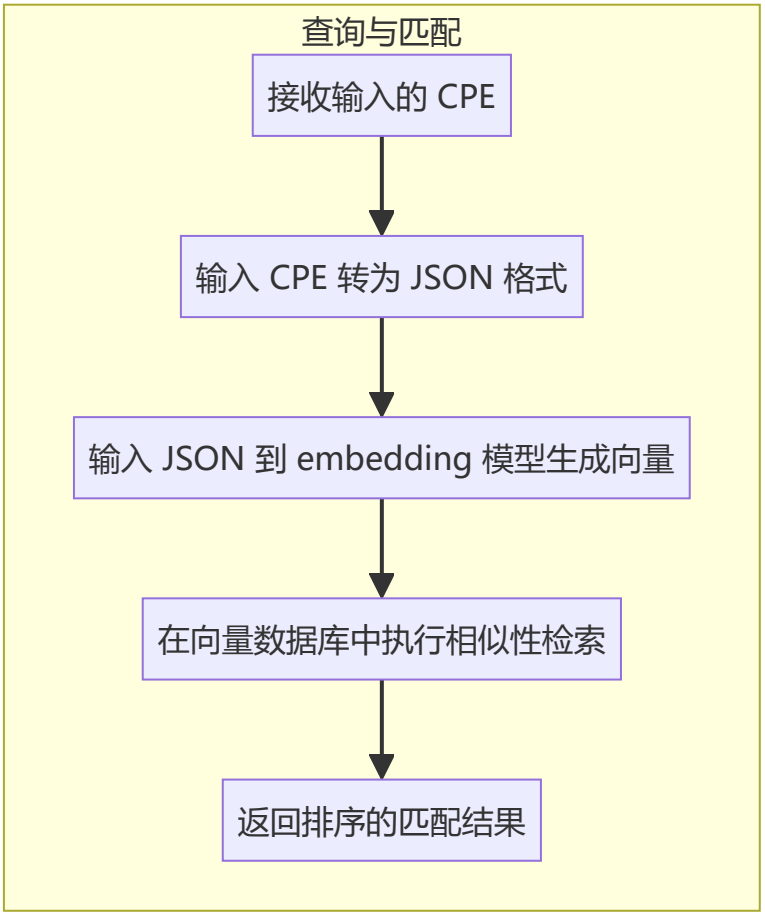
3.2. **JSON格式转换**：使用`cpe2json`方法将输入的CPE字符串转换为JSON格式。

3.3. **向量生成**：将JSON字符串输入Embedding模型，生成该CPE的向量表示。

3.4. **相似性检索**：在向量数据库中基于生成的向量进行相似性检索，找到最匹配的CPE项。

3.5. **结果排序和返回**：将检索结果按相似度排序，输出前几个最匹配的CPE项，并附上相似度得分。

流程图：



综合流程总结

1. **数据存储阶段**：从数据库中提取并转化CPE数据，将其向量化后存入向量数据库。
2. **大模型提取与格式化阶段**：从漏洞情报文本中提取符合CPE格式的信息，格式化为标准的JSON输出。
3. **查询与匹配阶段**：接收CPE输入后生成向量，在向量数据库中进行相似性检索，返回最匹配的结果。

该流程可以有效支持基于CPE的漏洞情报提取和相似性检索，为安全团队提供精确的匹配结果和数据支持。

二、附件

英文版本提示词：

```
You are a world-class cybersecurity expert specializing in the analysis and
handling of vulnerability intelligence. From the provided vulnerability
description, please extract all relevant network security CPEs (Common Platform
Enumeration) that adhere to the CPE format.

**Requirements for Extraction:**
```

```
1. The output should consist only of strings in the exact CPE format: `cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}`.
2. If certain fields are missing in the original CPE data, retain the details as they are without adding or modifying any information.

**Output Specifications:**

- Format the results as a JSON object with a single key, `cpe`, whose value is an array listing each extracted CPE string individually.
- No extra explanations or descriptions are required.

### Vulnerability Intelligence:
```

中文版本提示词：

```
你是一位全球顶尖的网络安全专家，专注于漏洞情报的分析和处理。请从提供的漏洞描述中提取所有符合CPE格式的网络安全CPE（通用平台枚举）。

**提取要求：**

1. 输出内容仅包含符合CPE标准的字符串，格式为：`cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}`。
2. 若某些CPE字段在漏洞情报中缺失，请严格遵循原始信息，不补充或修改任何内容。

**输出格式：**

- 将结果以JSON格式输出，显示为一个包含单一键`cpe`的JSON对象，其值为一个数组，数组中逐一列出每个提取的CPE字符串。
- 不需要添加任何额外的解释或说明。

### 漏洞情报：
```

优化后的 Prompt（中文版）

```
你是一位全球顶尖的网络安全专家，专注于漏洞情报的分析和处理。请从以下漏洞描述中提取所有符合 **CPE 标准** 的网络安全 CPE（通用平台枚举）。提取规则和输出格式如下：

---

### 提取规则

1. **严格遵循 CPE 标准格式：**
   CPE 字符串必须符合以下格式：

   ```
   cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}
   ```

2. **占位符填充：**
   如果漏洞描述中某些字段缺失，必须使用占位符 `*` 填充，确保每个字段都完整体现。

3. **提取条件：**
```

- 只提取 CPE 格式完全或部分匹配的信息。
- 不对原始数据进行猜测或推断，所有字段严格基于原始漏洞描述中的内容。

输出要求

1. **输出为 JSON 格式：**
   仅输出一个 JSON 对象，包含单一键名 `cpe`，其值为一个数组，数组内列出所有提取的 CPE 字符串。

2. **数组排序：**
   如果提取到多个 CPE 字符串，请按提取顺序列出，确保每个字符串独立展示。

3. **格式示例：**
   输入示例：

   ```json
   {
     "vuln_name": "Linux kernel 安全漏洞",
     "vuln_desc": "Linux kernel是美国Linux基金会的开源操作系统Linux所使用的内核。Linux kernel 存在安全漏洞，攻击者可利用该漏洞通过Frozen EBPF Map Race绕过Linux内核的限制进而提升权限。",
     "effect_scope": null
   }
   ```

   输出示例：

   ```json
   {
     "cpe": [
       "cpe:/o:linux:linux_kernel:*:*:*:*:*:*:*:*"
     ]
   }
   ```

   输入示例：

   ```json
   {
     "vuln_name": "某工控软件的漏洞",
     "vuln_desc": "Mitsubishi Electric 的MC Works64软件以及Iconics Genesis64软件均存在权限提升漏洞。",
     "effect_scope": "工业控制软件"
   }
   ```

   输出示例：

   ```json
   {
     "cpe": [
       "cpe:/a:mitsubishi_electric:mc_works64:*:*:*:*:*:*:*:*",
       "cpe:/a:iconics:genesis64:*:*:*:*:*:*:*:*"
     ]
   }
```

```
---
请基于上述规则提取并输出漏洞描述中的 CPE 字符串，确保格式一致，字段完整，无额外说明。

### 漏洞情报：
```

Optimized Prompt (英文版本)

```
You are a world-class cybersecurity expert specializing in the analysis and handling of vulnerability intelligence. Please extract all valid **CPE (Common Platform Enumeration)** strings from the following vulnerability description based on the rules and output format outlined below:

---

### Extraction Rules

1. **Strict Compliance with the CPE Standard Format:**
   CPE strings must strictly follow this format:

   ```
   cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}
   ```

2. **Placeholder Usage:**
   If certain fields are missing in the vulnerability description, they must be represented with the placeholder `*`, ensuring every field is fully included.

3. **Extraction Criteria:**

   - Only extract information that matches the complete or partial CPE format.
   - Do not infer or guess any fields; all fields must strictly align with the content in the original vulnerability description.

Output Requirements

1. **Output in JSON Format:**
   Provide a single JSON object with the key `cpe`. Its value should be an array, listing all extracted CPE strings.

2. **Array Order:**
   If multiple CPE strings are extracted, list them in the order they are identified, ensuring each string is presented independently.

3. **Format Examples:**
   Input Example:

   ```json
   {
     "vuln_name": "Linux kernel security vulnerability",
```
```

````
    "vuln_desc": "Linux kernel is the kernel used by the Linux operating system, an open-source project managed by the Linux Foundation. A security vulnerability exists in the Linux kernel, which attackers can exploit through the Frozen EBPF Map Race to bypass kernel restrictions and escalate privileges.",
    "effect_scope": null
  }
  ```

  Output Example:

  ```json
  {
    "cpe": [
      "cpe:/o:linux:linux_kernel:*:*:*:*:*:*:*:*"
    ]
  }
  ```

  Input Example:

  ```json
  {
    "vuln_name": "Industrial control software vulnerability",
    "vuln_desc": "Mitsubishi Electric's MC Works64 software and Iconics Genesis64 software both have privilege escalation vulnerabilities.",
    "effect_scope": "Industrial control software"
  }
  ```

  Output Example:

  ```json
  {
    "cpe": [
      "cpe:/a:mitsubishi_electric:mc_works64:*:*:*:*:*:*:*:*",
      "cpe:/a:iconics:genesis64:*:*:*:*:*:*:*:*"
    ]
  }
  ```

---

Please extract and output the CPE strings from the vulnerability description based on the above rules. Ensure consistent formatting, complete fields, and no additional explanations.

### Vulnerability Intelligence:
````
````