Caiss库说明文档

1. 简介

随着人工智能技术的普及，海量高维度向量的相似度查询技术在研究和生产中的作用和重要性与日俱增。目前，市面有许多优秀开源的解决方案，但是，在使用过程中发现了一些问题。比如：

- 由于对于各种算法原理的了解不深，不会调整参数，导致的训练模型结果偏差较大。
- 开源库对于各种距离的支持有限，无法满足随时变化的实验需求。
- 标签信息和向量的分离，导致标记和查询需要在不同的步骤中完成。
- 功能不够全面，很难一次性完成日常需要的"增删改查"功能。
- 部分解决方案，对于平台或者对于编程语言的依赖，导致了各种环境问题。

在这里，我们基于Google，Facebook，阿里巴巴等科技巨头的现有成果，实现了一套全新思路开源的解决方案。提供面向最终结果的训练方法，会在训练过程中，根据设定的目标自动调节参数。提供常用距离和自定义距离的训练和查询方式。支持训练过程中，标签信息和向量信息的绑定。支持缓存和多线程调用。提供纯C风格的SDK接口，同时支持SQL语法进行增删查改。支持Windows，Linux和Mac系统，方便迁移到其他编程语言（如Python，Java等），并提供了大量的Demo示例。

我们把这个库，命名为Caiss (Chunel Artificial Intelligence Similarity Search)。经过实测，它可以将原先100分钟才能暴力计算完成的逻辑，在保持97%准确率的情况下，耗时降低至20秒左右。且随着数据量的不断增加，其性能上的优势会更加明显。希望它可以在大家的研究和生产过程中，发挥积极的作用。

2. 使用流程

1，安装python3环境，安装TensorFlow-v1.11.0版本，安装bert-serving-server库和bert-serving-client库。注：以上两个库，暂时无法配合tf-v2.0.0或以上版本正常使用。如必须使用tf-v2.0.0或以上版本，请自行解决训练特征向量的问题，并参考/doc/demo_2500words_768dim.txt中格式，自行生成文件供caiss训练使用。

2，根据自身需求，下载对应的bert模型，并解压至本地。bert模型下载，请参考链接：bert模型下载说明

3，准备待embedding的文本文件。比如，英文单词的相似词查询任务，将不同的单词按行分开即可。格式请参考/doc/文件夹下的english-words-71290.txt文件。

4，执行/python/dataProcess/pyCaissTrainDataBuilder.py中下的**main**方法。执行前，需要根据实际情况，修改待embedding文本的位置（embedding_file_path），bert模型的位置（bert_model_path）。函数执行完毕后，会在result_path位置，生成可用于caiss库训练的文本内容。

5，参考下文第4部分关于Caiss的使用demo，开始训练、查询等功能吧。

3. 相关接口定义

```
/**
 * 初始化环境信息
 * @param maxThreadSize 支持的最大并发数
 * @param algoType 算法类型（详见CaissLibDefine.h文件）
 * @param manageType 并发类型（详见CaissLibDefine.h文件）
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
```

```
 */
CAISS_RET_TYPE CAISS_Environment(unsigned int maxThreadSize,
        CAISS_ALGO_TYPE algoType,
        CAISS_MANAGE_TYPE manageType);

/**
 * 创建句柄信息
 * @param handle 句柄信息
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_CreateHandle(void** handle);

/**
 * 初始化信息
 * @param handle 句柄信息
 * @param mode 处理类型（详见CaissLibDefine.h文件）
 * @param distanceType 距离类型（详见CaissLibDefine.h文件）
 * @param dim 维度
 * @param modelPath 模型路径
 * @param distFunc 距离计算函数（仅针对自定义距离计算生效）
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_Init(void *handle,
        CAISS_MODE mode,
        CAISS_DISTANCE_TYPE distanceType,
        unsigned int dim,
        const char *modelPath,
        CAISS_DIST_FUNC distFunc = nullptr);

/**
 * 模型训练功能
 * @param handle 句柄信息
 * @param dataPath 待训练样本路径（训练文件格式，参考/doc/文件夹下demo_2500words_768dim.txt的格式）
 * @param maxDataSize 最大样本个数
 * @param normalize 样本数据是否归一化
 * @param maxIndexSize 样本标签最大长度
 * @param precision 目标精确度
 * @param fastRank 快速查询排名个数
 * @param realRank 真实查询排名个数
 * @param step 迭代步径
 * @param maxEpoch 最大迭代轮数 （maxEpoch轮后，准确率仍不满足要求，则停止训练，返回警告信息）
 * @param showSpan 信息打印行数
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 * @notice 当快速查询fastRank个数，均在真实realRank个数的范围内的准确率，超过precision的时候，训练完成
 */
CAISS_RET_TYPE CAISS_Train(void *handle,
        const char *dataPath,
        unsigned int maxDataSize,
        CAISS_BOOL normalize,
        unsigned int maxIndexSize = 64,
        float precision = 0.95,
        unsigned int fastRank = 5,
        unsigned int realRank = 5,
```

```
        unsigned int step = 1,
        unsigned int maxEpoch = 5,
        unsigned int showSpan = 1000);

/**
 * 查询功能
 * @param handle 句柄信息
 * @param info 待查询的信息
 * @param searchType 查询信息的类型（详见CaissLibDefine.h文件）
 * @param topK 返回最近的topK个信息
 * @param filterEditDistance 需要过滤的最小词语编辑距离
 * @param searchCBFunc 查询到结果后，执行回调函数，传入的是查询到结果的word信息和distance信息
 * @param cbParams 回调函数中，传入的参数信息
 * @return 运行成功返回0，警告返回1，词查询模式下，没有找到单词返回2，其他异常值，参考错误码定义
 * @notice filterEditDistance仅针对根据单词过滤的情况下生效。
 *         =-1表示不过滤；=0表示过滤跟当前词语完全相同的；
 *         =3表示过滤跟当前词语相编辑距离的在3以内的，以此类推；
 *         最大值不超过CAISS_MAX_EDIT_DISTANCE值
 */
CAISS_RET_TYPE CAISS_Search(void *handle,
        void *info,
        CAISS_SEARCH_TYPE searchType,
        unsigned int topK,
        unsigned int filterEditDistance = CAISS_DEFAULT_EDIT_DISTANCE,
        CAISS_SEARCH_CALLBACK searchCBFunc = nullptr,
        const void *cbParams = nullptr);

/**
 * 获取结果字符串长度
 * @param handle 句柄信息
 * @param size 结果长度
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_GetResultSize(void *handle,
        unsigned int &size);

/**
 * 获取查询结果信息
 * @param handle 句柄信息
 * @param result 结果信息
 * @param size 对应结果长度
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_GetResult(void *handle,
        char *result,
        unsigned int size);

/**
 * 插入信息
 * @param handle 句柄信息
 * @param node 待插入的向量信息
 * @param label 待插入向量的标签信息
 * @param insertType 插入类型（详见CaissLibDefine.h文件）
```

```
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 * @notice 插入信息实时生效。程序结束后，是否保存新插入的信息，取决于是否调用CAISS_Save()方法
 */
CAISS_RET_TYPE CAISS_Insert(void *handle,
        CAISS_FLOAT *node,
        const char *label,
        CAISS_INSERT_TYPE insertType);

/**
 * 忽略信息
 * @param handle 句柄信息
 * @param label 待忽略的标签信息
 * @param isIgnore 表示忽略（true）或者不再忽略（false）
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_Ignore(void *handle,
        const char *label,
        CAISS_BOOL isIgnore = CAISS_TRUE);

/**
 * 保存模型
 * @param handle 句柄信息
 * @param modelPath 模型保存路径（默认值是覆盖当前模型）
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_Save(void *handle,
        const char *modelPath = nullptr);

/**
 * 执行sql指令
 * @param handle 句柄信息
 * @param sql 查询的sql语句
 * @param sqlCBFunc 执行sql过程中，触发的回调函数
 * @param sqlParams 传入的条件信息
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_ExecuteSQL(void *handle,
        const char *sql,
        CAISS_SQL_CALLBACK sqlCBFunc = nullptr,
        const void *sqlParams = nullptr);

/**
 * 销毁句柄信息
 * @param handle 句柄信息
 * @return 运行成功返回0，警告返回1，其他异常值，参考错误码定义
 */
CAISS_RET_TYPE CAISS_DestroyHandle(void *handle);
```

4. 使用Demo

```
/*
* 更多使用样例，请参考caissDemo文件夹中内容。
```

```
* 使用过程中，请对每个CAISS_*函数的返回值进行判断和处理。
* doc文件夹中，提供了demo_2500words_768dim.txt文件。
* 如果需要训练新的查询模型，请根据此文件的样式，生成新的词向量文件。
*/

#include <iostream>
#include <string>
#include "CaissLib.h"

using namespace std;

static const unsigned int max_thread_num_ = 1;    // 线程数量
static const CAISS_ALGO_TYPE algo_type_ = CAISS_ALGO_HNSW;
static const CAISS_MANAGE_TYPE manage_type_ = CAISS_MANAGE_SYNC;
static const CAISS_MODE mode_ = CAISS_MODE_PROCESS;
static const CAISS_DISTANCE_TYPE dist_type_ = CAISS_DISTANCE_INNER;
static const unsigned int dim_ = 768;    // 向量维度
static const char *model_path_ = "demo_2500words_768dim.caiss";
static const CAISS_DIST_FUNC dist_func_ = nullptr;
static const std::string info_ = "water";
static const CAISS_SEARCH_TYPE search_type_ = CAISS_SEARCH_WORD;
static const unsigned int top_k_ = 5;

static const string data_path_ = "demo_2500words_768dim.txt";
static const unsigned int max_data_size_ = 5000;    // 建议略大于训练样本中的行数，方便今后插入数据的更新
static const CAISS_BOOL normalize_ = CAISS_TRUE;    // 是否对数据进行归一化处理（常用于计算cos距离）
static const unsigned int max_index_size_ = 64;     // 标签的最大长度
static const float precision_ = 0.95;               // 模型精确度
static const unsigned int fast_rank_ = 5;
static const unsigned int real_rank_ = 5;
static const unsigned int step_ = 1;
static const unsigned int max_epoch_ = 3;
static const unsigned int show_span_ = 1000;

static int train() {
    /* 训练功能 */
    int ret = CAISS_RET_OK;

    void *handle = nullptr;
    ret = CAISS_CreateHandle(&handle);

    ret = CAISS_Init(handle, CAISS_MODE_TRAIN, dist_type_, dim_, model_path_, dist_func_);

    ret = CAISS_Train(handle, data_path_.c_str(), max_data_size_, normalize_,
max_index_size_,
            precision_, fast_rank_, real_rank_, step_, max_epoch_, show_span_);

    ret = CAISS_DestroyHandle(handle);
    return ret;
}
```

```
static int search() {
    /* 查询功能 */
    int ret = CAISS_RET_OK;

    void *handle = nullptr;
    ret = CAISS_CreateHandle(&handle);

    ret = CAISS_Init(handle, CAISS_MODE_PROCESS, dist_type_, dim_, model_path_,
dist_func_);

    ret = CAISS_Search(handle, (void *)info_.c_str(), search_type_, top_k_);

    unsigned int size = 0;
    ret = CAISS_GetResultSize(handle, size);

    char *result = new char[size + 1];
    memset(result, 0, size + 1);
    ret = CAISS_GetResult(handle, result, size);
    std::cout << result << std::endl;
    delete [] result;

    ret = CAISS_DestroyHandle(handle);

    return ret;
}

int main() {
    /* 使用过程中，请注意添加针对返回值ret的判定 */
    int ret = 0;
    ret = CAISS_Environment(max_thread_num_, algo_type_, manage_type_);

    ret = train();
    // ret = search();
    return 0;
}
```

5. 输出内容

- 训练接口执行完毕后，会在对应的目录下生成 *.caiss 模型文件。不同操作操作系统之间生成的模型文件，不能混用。如需跨平台使用，请重新训练。
- 查询结果输出，为标准json格式。例：查询词语water，查询topK=5，返回相似词语为：[water,wine,mud,food,soup]这5个词语，具体结果信息如下：

```
{
    "version":"1.2.0",
    "size":5,
    "distance_type":"inner",
    "search_type":"ann_search",
    "details":[
        {
            "distance":0,
```

```
            "index":287,
            "label":"water"
        },
        {
            "distance":0.07434636354446411,
            "index":3102,
            "label":"wine"
        },
        {
            "distance":0.10038524866104126,
            "index":6950,
            "label":"mud"
        },
        {
            "distance":0.10039275884628296,
            "index":641,
            "label":"food"
        },
        {
            "distance":0.10307157039642334,
            "index":7153,
            "label":"soup"
        }
    ]
}
```


6. 编译说明

- 本人在Windows（Win10），Linux（Ubuntu-16.04）和Mac(MacOS-10.15)上开发，使用的IDE均是CLion。编译依赖boost库，本人的库是boost-1.67.0。建议使用不低于此版本的boost库，以免出现编译问题。
- Linux命令行模式下，进入caiss文件夹下（与CMakeList.txt同级目录），输入：
  $ cmake .

  $ make
  即可完成编译（前提：环境中支持cmake命令）。

7. 补充说明

- 训练文本样式，请参考文档中的内容
- 训练功能仅支持单线程。查询和插入功能，支持多线程并发
- 新增数据实时生效。进程重启后是否生效，取决于是否调用save方法
- 在异步模式下，插入、查询等需要传入向量信息的方法中，请自行保证传入的向量数据（内存）持续存在，直到获取结果为止
- doc文件夹中，提供了供测试使用的2500个常见英文单词的词向量（768维）文件，仅作为本库的测试样例使用，有很多常见的词语都没有包含，更无任何效果上的保证。如果需要完整的词向量文件，请自行训练，或者联系微信：Chunel_Fung
- 本库的源代码，发布在：https://github.com/ChunelFeng/caiss 。欢迎随时交流指导

8. 版本信息

[2020.06.15 - v1.0.0 - Chunel]

- 新建，第一个功能版本
- 实现训练、查询、插入、保存等功能

[2020.06.25 - v1.2.0 - Chunel]

- 新增多线程功能
- 新增缓存机制
- 提供简单的使用demo
- 兼容mac和Linux系统

[2020.06.29 - v1.2.1 - Chunel]

- 异步并发模式优化

[2020.07.03 - v1.3.0 - Chunel]

- 新增根据编辑距离的过滤
- 新增python版本，提供基础查询功能

[2020.07.11 - v1.5.0 - Chunel]

- 新增词语删除功能
- 新增并行计算功能，极大缩短训练和验证耗时
- 基于python版本，对外提供网络服务

[2020.07.18 - v1.5.1 - Chunel]

- 优化异步查询过程中，查询单词信息内存自动释放的问题

[2020.08.01 - v1.5.2 - Chunel]

- mac版本中，提供并行计算方法，进一步减少查询耗时
- 解决跨平台兼容性问题

[2020.08.22 - v1.6.0 - Chunel]

- 加入降维算法。确保少量降低准确率的情况下，大幅度降低查询耗时
- 提供针对文本的embedding处理方法，方便自行生成符合格式的训练文件

[2020.08.30 - v2.0.0 - Chunel]

- 提供基础SQL语句查询功能

[2020.09.12 - v2.0.1 - Chunel]

- 提供基于SQL语句的增删查改功能
- 提供SQL版本的使用demo