Uživatelská dokumentace

Tato dokumentace obsahuje návod jak spustit ukázku nové implementace indexu HNSW, která je součástí bakalářské práce na téma "Aproximace KNN problému". Také obsahuje podrobný návod ke všem programům, které obsahuje příloha práce.

Všechny cesty uvedené v tomto souboru jsou relativní k cestě složky, která obsahuje PDF soubor této dokumentace.

Potřebné programy

- Docker
- Překladač C++17
- Python 3.9

Ukázka

1. Ujistěte se, že služba Docker je zapnutá.
2. Spusťte skript `RUNME.py` pomocí interpretu Python verze 3.9. Na Windows například takto:

```
py -3.9 RUNME.py
```

Tento skript provede následující.

- Vytvoří virtuální prostředí interpretu Python ve složce `.venv`.
- Vytvoří nativní C++ řešení ve složce `src/cmakeBuild`.
- Spustí porovnání původní a nové implementace indexu HNSW nad malými kolekcemi se 100 000 prvky.
- Otevře stránku s výsledky v jedné kartě internetového prohlížeče.
- Otevře stránku s dokumentací indexu ve druhé kartě.

Výsledky srovnání

Výsledky jsou zaznačeny do grafů, které zobrazuje webová stránka. Ta je generována ve složce `src/website`. Původní implementace je v grafech označována slovem `original`, nová implementace slovem `new`.

Virtuální prostředí

Pokud není uvedeno jinak, skripty uvnitř složky `src/scripts` vždy spouštějte pomocí vygenerovaného virtuální prostředí. Prostředí aktivujete pomocí aktivačního skriptu ve složce `.venv/Scripts`. Výběr skriptu závisí na použitém OS a interpretu.

OS	INTERPRET	CESTA K AKTIVAČNÍMU SKRIPTU
Linux		./.venv/Scripts/activate
Windows	Batch	.\.venv\Scripts\activate.bat
Windows	Powershell	.\.venv\Scripts\Activate.ps1

Seznam skriptů

Následuje seznam skriptů ve složce `src/scripts`, které umožňují uživateli provést více operací než úvodní skript.

NÁZEV SKRIPTU	STRUČNÝ POPIS SKRIPTU
buildProject	Vytvoří virtuální prostředí, nativní C++ řešení a jeho Python rozhraní.
clean	Odstraní vygenerované soubory a vrátí projekt do původního stavu.
datasetGenerator	Vygeneruje datové soubory pro debugování.
datasetToText	Převede datový soubor do textového formátu.
formatCMakeTemplates	Vygeneruje CMakeLists.txt.
generateTables	Vygeneruje LaTeX tabulky podobné těm, které jsou v bakalářské práci.
latexTable	Vygeneruje LaTeX tabulku na základě výsledků srovnání.
runBenchmarks	Spustí srovnání, vygeneruje a otevře webovou stránku s výsledky.
runRecallTable	Postaví nový index a zobrazí tabulku závislosti přesnosti na parametru vyhledávání ef_{search}.

NÁZEV SKRIPTU	STRUČNÝ POPIS SKRIPTU
SIMDCapability	Zobrazí SIMD rozšíření instrukční sady procesoru, která jsou k dispozici.

Podrobný popis skriptů

U každého skriptu je uveden jeho účel, parametry a příklad spuštění. Pokud skript obsahuje alespoň jeden parametr, pak použitím parametru `--help` nebo `-h` zobrazíte nápovědu v anglickém jazyce.

buildProject

Tento skript lze spustit bez virtuálního prostředí.

Vytvoří virtuální prostředí interpretu Python, stáhne potřebné softwarové balíčky, vygeneruje nativní C++ řešení pro knihovnu nového indexu, vytvoří rozhraní v jazyce Python pro nový index a otestuje funkčnost tohoto indexu spuštěním skriptu `runRecallTable`.

PARAMETR, ZKRATKA	VÝZNAM
--clean, -c	Vrátí projekt do původního stavu před jeho opětovným sestavením.
--cleanResults, -r	Pokud je `--clean` nastaven, odstraní naměřené výsledky.
--ignorePythonVersion, -i	Umožňuje spustit skript s libovolnou verzí interpretu Python. Skript poté nemusí fungovat správně.

Příklad spuštění:

```
py -3.9 buildProject.py --clean --cleanResults
```

clean

Tento skript lze spustit bez virtuálního prostředí.

Odstraní datové soubory pro debugování, C++ nativní řešení a Python rozhraní. Pokud je spuštěn mimo virtuální prostředí, pak odstraní toto prostředí. Naměřené výsledky odstraněny nebudou, pokud o to uživatel nepožádá.

PARAMETR, ZKRATKA	VÝZNAM
--results, -r	Odstraní naměřené výsledky srovnání, vygenerované grafy a tabulky.

Příklad spuštění:

```
py clean.py --results
```

datasetGenerator

Vygeneruje datové soubory pro debugování uvedené v konfiguračním souboru `src/config/debugDatasets.json`. O konfiguraci tohoto skriptu se více dočtete v kapitole `Datové soubory pro debugování` níže v této dokumentaci.

Příklad spuštění:

```
py datasetGenerator.py
```

datasetToText

Převede vybraný datový soubor ze složky `src/data` do textového formátu. Výstupní textový soubor zapíše pod jménem datového souboru do stejné složky.

PARAMETR, ZKRATKA	VÝZNAM
--name, -n	Název datového souboru bez přípony. Pokud není uveden, výchozím souborem je `angular-small`.

Příklad spuštění:

```
py datasetToText --name euclidean-medium
```

formatCMakeTemplates

Vygeneruje soubor `src/index/CMakeLists.txt` a doplní do něj správnou definici maker tak, aby došlo pouze ke kompilaci těch funkcí, pro které je k dispozici vhodné SIMD rozšíření instrukční sady procesoru.

Příklad spuštění:

```
py formatCMakeTemplates.py
```

generateTables

Vygeneruje LaTeX tabulky podobné těm, které jsou v bakalářské práci, ale pouze v případě, že jsou pro ně dostupné naměřené výsledky. Tyto výsledky lze získat spuštěním následujících příkazů. Avšak tato měření mohou trvat více než 12 hodin.

```
py runBenchmarks.py -a ..\config\heuristic.yaml ..\config\naive.yaml -d lastfm-64-dot -r 5
py runBenchmarks.py -a ..\config\heuristic.yaml ..\config\prefetch.yaml -d glove-50-angular -r 5
py runBenchmarks.py -a ..\config\original.yaml ..\config\prefetch.yaml -d sift-128-euclidean -r 5
py generateTables.py
```

Vygenerované tabulky jsou dostupné ve složce `src/figures`.

latexTable

Vygeneruje jednu LaTeX tabulku na základě výsledků srovnání implementací.

PARAMETR, ZKRATKA	VÝZNAM
--algorithms, -a	Vyžadován. Seznam implementací oddělený mezerami.
--dataset, -d	Vyžadován. Název datového souboru.
--label, -la	Identifikátor tabulky.
--legend, -le	Názvy implementací v tabulce. Pokud není uveden, budou použity původní názvy.
--output, -o	Vyžadován. Cesta k výstupnímu souboru.
--percent, -p	Přidá do tabulky sloupec s procentuálním rozdílem časů stavby.
--recompute, -r	Znovu vypočítá výkonnostní metriky z naměřených výsledků. Tato operace může trvat více než 10 minut.

Příklad spuštění:

```
py latexTable.py -a new-prefetch original -d sift-128-euclidean -le "Nová impl." "Původní impl." -o ..\figures\table.tex -p
```

runBenchmarks

Před spuštěním se ujistěte, že je služba Docker zapnutá.

Spustí srovnání implementací v jednom nebo více Docker kontejnerech, vypočítá výkonnostní metrika, vygeneruje webovou stránku s výsledky a otevře ji v nové kartě internetového prohlížeče. Kód vygenerované stránky lze poté najít ve složce `src/website` a můžete ji opětovně zobrazit otevřením souboru `index.html`.

PARAMETR, ZKRATKA	VÝZNAM
--algoDefPaths, -a	Vyžadován. Seznam cest ke konfiguračním souborům oddělených mezerami. O konfiguraci se více dočtete v kapitole `Konfigurace srovnání`.
--datasets, -d	Vyžadován*. Seznam datových souborů oddělených mezerami.
--datasetsPath, -p	Vyžadován*. Cesta k textovému souboru se seznamem datových souborů.
--force, -f	Spustí již provedená měření znovu.-
--runs, -r	Počet opakování měření. Výchozí hodnota je 1.
--workers, -w	Počet paralelně spuštěných Docker kontejnerů. Výchozí hodnota je 1.

Datové soubory využité ke srovnání nejsou ty samé, které jsou využívány k debugování. Jejich seznam najdete v kapitole `Testované datové soubory`.

* Pouze jeden z parametrů označených hvězdičkou by měl být uveden.

Příklad spuštění:

```
py runBenchmarks.py -a ..\config\noBit.yaml -f -p
..\config\datasets.txt -r 5 -w 2
```

runRecallTable

Postaví index nové implementace a vyhledá v něm nejbližší sousedy s různými hodnotami parametru vyhledávání ef_{search}. Poté vypíše tabulku závislosti přesnosti vyhledávání na tomto parametru. Konfigurace tohoto skriptu se nachází v souboru `src/config/recallTable.json` a více se o ní dočtete v kapitole `Konfigurace programů recallTable`.

Příklad spuštění:

```
py runRecallTable.py
```

SIMDCapability

Zobrazí SIMD rozšíření instrukční sady procesoru, která jsou k dispozici. Využívána ostatními skripty pro vygenerování správných maker v jazyce C++.

Příklad spuštění:

```
py SIMDCapability.py
```

Nativní knihovna

C++ řešení vygenerujete pomocí skriptu `RUNME.py` nebo `src/scripts/buildProject.py`. Řešení bude vytvořeno ve složce `src/cmakeBuild`. V každém systému vypadají soubory řešení jinak. Např. při použití Windows s Visual Studiem je řešením `.sln` soubor a projekty jsou `.vcxproj` soubory. Pro spuštění projektů je doporučena konfigurace `Release`. Řešení obsahuje dva projekty.

- *datasetToText* - Vypíše textový popis datové kolekce do souboru. Slouží pro ověření konzistence mezi binárními a HDF5 soubory. Název datového souboru je prvním parametrem programu. Výchozí hodnotou je `angular-small`.
- *recallTable* - Postaví HNSW index a vypíše tabulku závislosti přesnosti na parametru vyhledávání ef_{search}. Konfigurace programu se nachází v souboru `src/config/recallTable.json` a více se o ní dočtete v kapitole `Konfigurace programů recallTable`.

Datové soubory pro debugování

Pro vygenerování jiných datových souborů pro debugování změňte konfiguraci v souboru `src/config/debugDatasets.json` a spusťte skript `src/scripts/datasetGenerator.py`. Konfigurace je JSON soubor s polem objektů, kde každý objekt popisuje jeden datový soubor.

```json
{
    "name": "angular-small",
    "angular": true,
    "dim": 25,
    "k": 10,
    "testCount": 200,
    "trainCount": 20000,
    "seed": 104
}
```

KLÍČ	TYP HODNOTY	VÝZNAM
name	string	Unikátní název souboru sloužící k identifikaci.
angular	boolean	Pokud je nastaven na `true`, využívá soubor kosinusové podobnosti. Jinak využívá Eukleidovské vzdálenosti.
dim	int	Počet dimenzí prostoru.
k	int	Počet hledaných nejbližších sousedů dotazovaného prvku.
testCount	int	Počet dotazů.
trainCount	int	Počet prvků použitých k sestavení indexu.
seed	int	Nastavení generátoru náhodných čísel.

Konfigurace programů *recallTable*

Pro změnu datového souboru nebo nastavení indexu v programech `recallTable.cpp` a `recallTable.py` upravte soubor `src/config/recallTable.json`. Konfigurace je JSON soubor s jediným objektem.

```json
{
    "dataset": "angular-small",
    "efConstruction": 200,
    "efSearch": [10, 15, 20, 40, 80, 120, 200],
    "mMax": 16,
    "seed": 200,
    "SIMD": "best",
    "template": "prefetching"
}
```

KLÍČ	TYP HODNOTY	VÝZNAM
dataset	string	Identifikace datového souboru. Odpovídá klíči `name` v souboru `src/config/debugDatasets.json`.
efConstruction	int	Počet uvažovaných sousedů při vytváření nových hran v indexu.
efSearch	array	Pole hodnot parametru vyhledávání ef_{search}.
mMax	int	Maximální povolený počet sousedů jednoho prvku v indexu na vrstvě vyšší než vrstva 0.
seed	int	Nastavení generátoru náhodných úrovní v indexu.
SIMD	string	Upřednostňovaný typ SIMD instrukcí. Možnosti jsou `avx`, `avx512`, `best`, `null`, a `sse`.*
template	string	Šablona indexu. Možnosti jsou `Heuristic`, `Naive`, `NoBitArray` a `Prefetching`.

* Zvolením hodnoty `best` zvolíte nejmodernější dostupné SIMD rozšíření. Hodnotou `null` zakážete použití SIMD instrukcí.

Šablony nové implementace

ŠABLONA	METODA VÝBĚRU SOUSEDŮ	SEZNAM NAVŠTÍVENÝCH VRCHOLŮ	ASYNCHRONNÍ PŘÍSTUP DO PAMĚTI
Heuristic	Heuristika	Bitové pole	Ne
Naive	Naivní algoritmus	Bitové pole	Ne
NoBitArray	Heuristika	Obyčejné pole	Při výpočtu vzdáleností Při načítání dat seznamu navštívených vrcholů
Prefetching	Heuristika	Bitové pole	Při výpočtu vzdáleností

Testované datové soubory

Pro srovnání implementací je možno využít následujících datových souborů.

NÁZEV	DIMENZE	POČET PRVKŮ PŘI STAVBĚ	DOTAZY	METRIKA

NÁZEV	DIMENZE	POČET PRVKŮ PŘI STAVBĚ	DOTAZY	METRIKA
deep-image-96-angular	96	9 990 000	10 000	Kosinusová podobnost
fashion-mnist-784-euclidean	784	60 000	10 000	Eukleidovská vzdálenost
gist-960-euclidean	960	1 000 000	1 000	Eukleidovská vzdálenost
glove-25-angular	25	1 183 514	10 000	Kosinusová podobnost
glove-50-angular	50	1 183 514	10 000	Kosinusová podobnost
glove-100-angular	100	1 183 514	10 000	Kosinusová podobnost
glove-200-angular	200	1 183 514	10 000	Kosinusová podobnost
lastfm-64-dot	64	292 385	50 000	Kosinusová podobnost
mnist-784-euclidean	784	60 000	10 000	Eukleidovská vzdálenost
nytimes-16-angular	16	290 000	10 000	Kosinusová podobnost
nytimes-256-angular	256	290 000	10 000	Kosinusová podobnost
random-s-100-angular	100	100 000	10 000	Kosinusová podobnost
random-s-100-euclidean	100	100 000	10 000	Eukleidovská vzdálenost
random-xs-20-angular	20	10 000	10 000	Kosinusová podobnost
random-xs-20-euclidean	20	10 000	10 000	Eukleidovská vzdálenost
sift-128-euclidean	128	1 000 000	10 000	Eukleidovská vzdálenost

Pro spuštění srovnání nad více soubory lze využít textového formátu, kde každý řádek reprezentuje jeden datový soubor. Řádky, které začínají znakem `#` jsou ignorovány.
Příklad:

```
# deep-image-96-angular
glove-25-angular
sift-128-euclidean
```

Konfigurace srovnání

Výběr implementací ke srovnání a jejich parametrů zprostředkovávají konfigurační soubory ve formátu YAML. Příklad takového souboru je `src/config/algos.yaml`. Ve složce `src/config` se nacházejí předem vytvořené konfigurace. Jejich význam popisuje následující tabulka.

NÁZEV SOUBORU	VÝZNAM
100k-large.yaml	12 stejných konfigurací pro novou a původní implementaci. Vhodné pro malé datové soubory.
100k-small.yaml	4 stejné konfigurace pro novou a původní implementaci. Vhodné pro malé datové soubory.
algos.yaml	1 konfigurace pro každou šablonu nové a původní implementace.
heuristic.yaml	12 konfigurací pro šablonu *Heuristic* nové implementace.*
naive.yaml	12 konfigurací pro šablonu *Naive* nové implementace.*
noBit.yaml	12 konfigurací pro šablonu *NoBitArray* nové implementace.*
original.yaml	12 konfigurací pro původní implementaci.
prefetch.yaml	12 konfigurací pro šablonu *Prefetching* nové implementace.*

* Popis šablon obsahuje kapitola `Šablony nové implementace`.

Následuje příklad konfigurace. Komentáře označené znakem `#` popisují jednotlivé klíče.

```
algos: # Povinný klíč.
  # Povinné pole implementací.
  original: # Název implementace.
    # Povinné pole konfigurací stavby.
    - efConstruction: 50 # Povinný klíč parametru efConstruction.
      mMax: 4 # Povinný klíč parametru mMax.
    - efConstruction: 50
      mMax: 8
    - efConstruction: 50
      mMax: 12
```

```
  new-prefetch:
    - efConstruction: 50
      mMax: 8
    - efConstruction: 50
      mMax: 12
efSearch: [10, 12, 14, 16, 18, 20, 25, 30, 40, 80] # Povinné pole hodnot parametru vyhledávání efSearch.
```

Definované implementace popisuje následující tabulka.

NÁZEV IMPLEMENTACE	DRUH IMPLEMENTACE	ŠABLONA	SIMD ROZŠÍŘENÍ
original	Původní hnswlib		Nejmodernější dostupné
new-avx	Nová	Heuristic	AVX
new-heuristic	Nová	Heuristic	Nejmodernější dostupné
new-naive	Nová	Naive	Nejmodernější dostupné
new-no-bit	Nová	NoBitArray	Nejmodernější dostupné
new-no-simd	Nová	Heuristic	Žádné
new-prefetch	Nová	Prefetching	Nejmodernější dostupné
new-sse	Nová	Heuristic	SSE