Vector Space Retrieval

An efficient Vector Space Model (VSM) implementation for retrieving medical data, built within the team project for the course Information Retrieval and Web Search.

Table of Contents

Problem Description

Describe the problem we want to solve in more detail here. Maybe just copy paste from report..

Prerequisites

There are several tools you will need to install and execute the application. In the following is a list with all required tools and technologies required for installing and running the system:

- ***UNIX-like*** operating system: All development and testing was carried out on UNIX-like operating systems. On ***Windows*** we encountered several compiler problems. Although fixed, we still strongly recommend to use a UNIX-like operating system (tested under ***macOS High Sierra Version 10.13.4***, Arch Linux [Version..] and Ubuntu [Version...])
- CMake: We use CMake as platform independent build system. Before compiling several requirements will be checked by CMake and the compilation process will either fail or a warning will be emitted if requirements are not satisfied.
- GNU Make: As development and testing was carried out on UNIX-like operating systems, the de-facto standard build and install system ***GNU Make*** is used internally by CMake.
- *C++17* compatible compiler: As all tests were carried out with ***GCC 7.3.0***, we recommend to use this one or a never version
- *boost*: We use the popular C++ library ***boost*** for several convenience functionalities such as efficiently splitting strings or dynamically creating bit vectors. If not already installed on your system, you will need to download the ***Header-Only Library*** part of ***boost***.
- Node.js: If you want to use the system with a web interface you will need to have ***Node.js*** installed. (more details alex..)
- Other libraries used in our project are the Oleander Stemming Library and a C++ JSON Library. The source code of these libraries is automatically cloned when running our installation script.

System Requirements

The retrieval system stores a lot of index structures in-memory. Depending on the settings and execution mode, the system might need up to 2GB of main memory.

Project Structure

data

All data files required for the system are stored here. We used the medical dataset nfcorpus from the Statistical NLP Group of the University of Heidelberg (a detailed description of the dataset can be found in data/README.md). By default, the system will search here for the required data files. Optionally, you can provide your own file paths as command line argument (for more details see Command Line Arguments). **Note**: For queries, it is not possible to provide a custom path for each query type but you can provide a path to the directory where all the query files are stored. It is **IMPORTANT** to keep the same naming conventions otherwise the query files can not be found (The convention is: `q-[Query Type].queries`; Possible Query Types are `{'all', 'nontopictitles', 'titles', 'viddesc', 'vidtitles'}`). Also, in our setting, the query document file uses a `'~'` delimiter between the query ID and its content. The delimiter is hard coded into the system and must therefore match.

docs

In this directory you can find the documentation of our source code. It was generated by using `doxygen`.

evsr-web

In order to use the system from within a web interface, we implemented a small web server. For further informations read the instructions in this directory.

python

Python scripts used for preprocessing and crawling the web.

src

Directory of the C++ source code of the system and its libraries. For further information regarding the source code, take a look into the Documentation at docs.

test

Directory of the unit tests.

Getting Started

The build and installation process will be described in the following. Follow the Quick Start Guide for a fast installation and get the system running. This works only if the *boost* library can be located in its default path. For a more detailed installation guide or if you encounter problems, take a look at Detailed Installation Guide.

Quick Start Guide

1. Make sure all the Requirements are satisfied

2. Clone the source with `git`:

```
git clone https://github.com/WeberNick/vector-space-retrieval.git
```

3. Build and install:

```
cd vector-space-retrieval
./install.sh
```

4. Run

```
./bin/evsr_run       # Run the system
./bin/Unit_Tests_run # Run the unit tests
```

Detailed Installation Guide

- Make sure all the Requirements are satisfied
- Clone the source with `git`:

```
git clone https://github.com/WeberNick/vector-space-retrieval.git
```

Set Up

- To install and build the system, several additional options can be (and sometimes must be) provided to the installation process. A complete list of options is provided later.
- If *boost* is not located in the default search path (on Linux `/usr/include/boost`, `/usr/local/Cellar/boost` if installed using *homebrew*), you have to provide the absolute or relative path to the include directory of *boost* via the `-b`option.
- If *GCC* is not the default compiler on your system (on macOS `g++` is often mapped to use `clang` instead) you may want to provide a path to the C++ compiler via the `-cxx` option (e.g., `/usr/bin/g++`)

```
cd vector-space-retrieval
./install.sh [-b [/path/to/boost/include]] [-cxx [/path/to/c++-compiler]]
```

Command Line Argument	Description	Default	Expects parameter
-cxx/--cxx	Set a custom C++ compiler path	Empty (Use system default)	String Path
-b/--boost	Set a path to Boost include dir	Empty (Try standard paths)	String Path
-a/--all	Deletes every generated directory as well as all cloning all external Libraries	false	-

Run

- To run the system, several additional command line arguments can be provided to the executable. A complete list of command line arguments is provided in the console by running

```
./bin/evsr_run --help
```

Command Line Arguments

In the following table we briefly introduce all the command line arguments.

Command Line Argument	Description	Default	Expects parameter
--help	Print all command line arguments	false	-
--trace	Activate tracing	false	-
--server	Start the binary in server mode	false	-
--measure	Activate the performance measurement	false	-
--collection-path	Path to the collection file	./data/d-collection.docs	String Path
--query--path	Path to the query directory	./data/	String Path
--scores-path	Path to the relevance score file	./data/s-3.qrel	String Path
--stopword-path	Path to the stopword file	./data/stopwords.large	String Path
--word-embeddings	Path to the word embeddings file	./data/w2v/glove.6B.300d.txt	String Path
--trace-path	Path to the log directory	./	String Path
--eval-path	Path to the evaluation directory	./	String Path
--topk	The top K results returned	20	unsigned int
--tiers	Number of tiers used by the tiered index	50	unsigned int
--dimensions	Number of dimensions used by the random projections	1000	unsigned int
--seed	Seed, used for random projections and cluster leader election	1	unsigned int