# ltr++

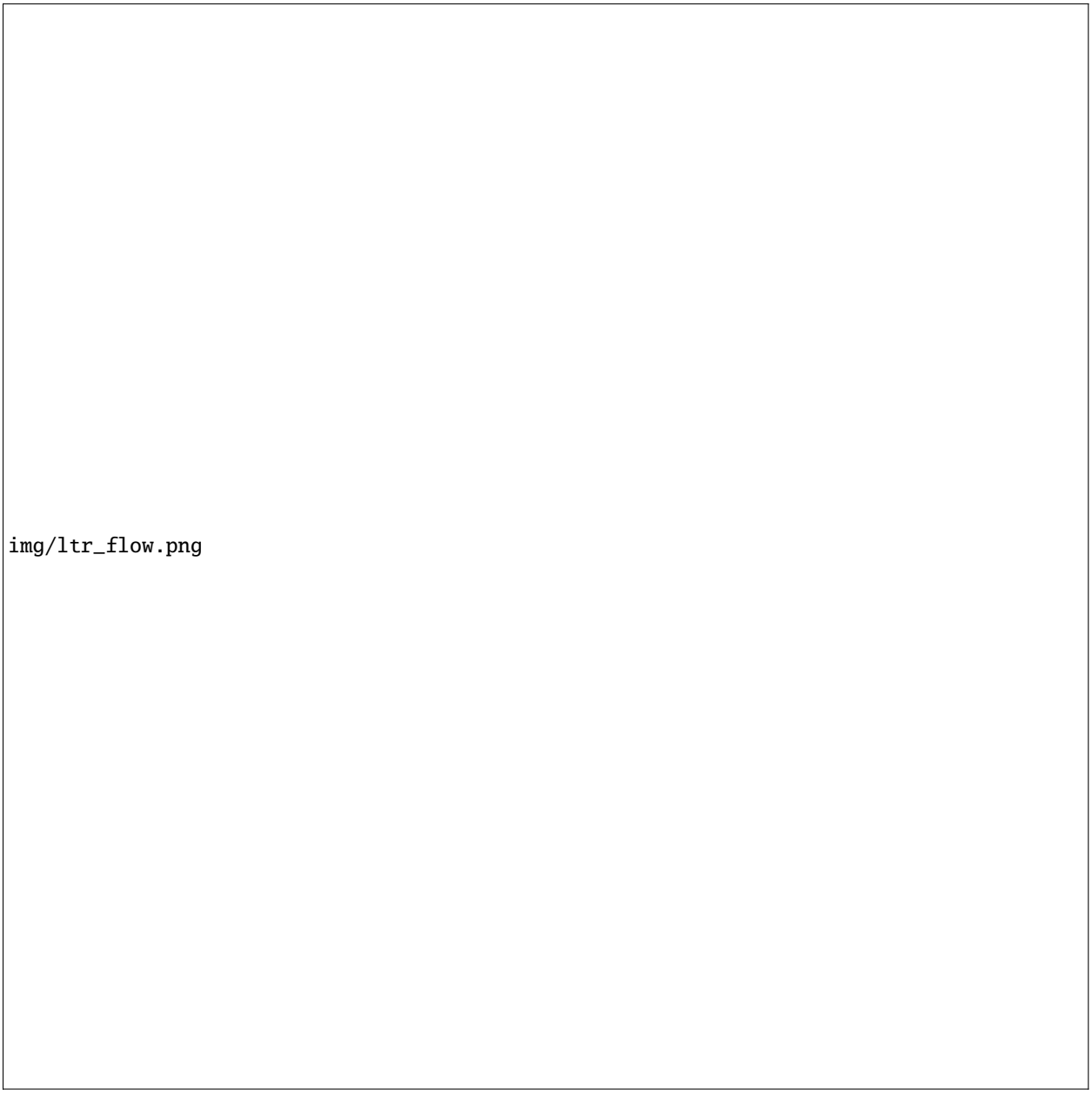
*Release 2021*

**Marcos Pontes and The Lemur Project**

**Oct 29, 2021**

# CONTENTS:

Learning to Rank (LTR) is one of the methods that can be effectively applied to solve the task of creating a ranking model in Information Retrieval. It helps solving IR problems such as document retrieval, collaborative filtering, sentiment analysis, computational advertising etc. LTR method aims at learning a model that given a query and a set of candidate documents finds the appropriate ranking of documents according to their relevancy. LTR method aims at learning a model that given a query and a set of candidate documents finds the appropriate ranking of documents according to their relevancy. **LTR++** is a Learning to Rank open-source library based on RankLib project. Currently two popular algorithms have been implemented and it also implements many retrieval metrics.




The LTR++ project is in early steps of development. Feel free to contribute or suggest features!

CHAPTER

ONE

# INTEGRATION

## 1.1 Prerequisites

The main prerequisite for installing ltr++ is CMake 3.15 or higher. Other external libraries will be downloaded automatically, if you don't have they. So that make sure you have CMake installed. Also, make sure that your operating system is compatible with any build that is working:

| System | 1.0.0 |
|---|---|
| Windows x86 | |
| Windows x64 | |
| MacOSX x64 | |
| Linux x64 | |

## 1.2 1. C++

Using CMake and certifying that all prerequisites are ok, let's understand how to build and install ltr.

### 1.2.1 Including header-only

After installing the ltr library, you can import the library as a header-only file:

```
#include <ltr.hpp>
```

**Warning:** Make sure you have C++17 or higher installed.

### 1.2.2 Embed as CMake subdirectory

You can use ltr++ directly in CMake projects as a subproject.

Clone the whole project inside your own project:

```
$ git clone https://github.com/marcosfpr/ltrpp/
```

and add the subdirectory to your CMake script:

```
add_subdirectory(ltrpp)
```

When creating your executable, link the library to the targets you want:

```
add_executable(my_target main.cpp)
target_link_libraries(my_target PRIVATE ltr)
```

Your target will be able to see the ltr++ headers now.

### 1.2.3 Embed with CPM.cmake

CPM.cmake is a nice wrapper around the CMake FetchContent function. Install CPM.cmake and then use this command to add ltr++ to your build script:

```
CPMAddPackage(
        NAME ltr
        GITHUB_REPOSITORY marcosfpr/ltrpp
        GIT_TAG origin/master # or whatever tag you want
)
# ...
target_link_libraries(my_target PUBLIC ltr)
```

Your target will be able to see the ltr++ headers now.

### 1.2.4 Find as CMake package

If you are using CMake and have the library installed on your system, you can then find ltr++ with the usual `find_package` command:

```
find_package(ltr REQUIRED)
# ...
target_link_libraries(my_target PUBLIC ltr)
```

Your target will be able to see the ltr++ headers now.

> **Warning:** There is no easy default directory for `find_package` on windows. You have to set it yourself.

## 1.3 2. Python

Development stage

## 1.4 3. Installing

Development stage

## 1.5 4. Building

### 1.5.1 4.1 Update C++

Update your C++ compiler to at least C++17:

**Ubuntu:**

```
# install GCC10
sudo apt install build-essential
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt install gcc-10
sudo apt install g++-10
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 10
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-10 10
# Choose gcc-10 there as the default compiler
update-alternatives --config g++
```

**Mac OS:**

```
# Download clang
curl --output clang.tar.xz -L https://github.com/llvm/llvm-project/releases/download/
↪llvmorg-11.0.0/clang+llvm-11.0.0-x86_64-apple-darwin.tar.xz
mkdir clang
tar -xvJf clang.tar.xz -C clang
# Copy the files to use/local
cd clang/clang+llvm-11.0.0-x86_64-apple-darwin
sudo cp -R * /usr/local/
# Make it your default compiler
export CXX=/usr/local/bin/clang++
```

**Windows:**

Update your Visual Studio Compiler

### 1.5.2 4.2 Building the projet

After installing or updating the dependencies, clone the project with:

```
git clone https://github.com/marcosfpr/ltrpp.git
cd pareto
```

And then build with:

**Ubuntu:**

```
mkdir build
cd build
cmake -version
cmake .. -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS="-O2"
cmake --build . -j 2 --config Release
# The next command for installing
sudo cmake --install .
# The next command for building the packages / installers
sudo cpack .
```

**Mac OS:**

```
mkdir build
cd build
cmake -version
cmake .. -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS="-O2"
cmake --build . -j 2 --config Release
# The next command for installing
cmake --install .
# The next command for building the packages / installers
cpack .
```

**Windows:**

```
mkdir build
cd build
cmake -version
cmake .. -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS="/O2"
cmake --build . -j 2 --config Release
# The next command for installing
cmake --install .
# The next command for building the packages / installers
cpack .
```

### 1.5.3 Python

Development stage

CHAPTER

TWO

# LICENSE

CHAPTER

# THREE

# NEED HELP?

Please contact marcos.rezende@aluno.ufop.edu.br or mfprezende@gmail.com