Indri Project Report

whuang022nccu

May 22, 2019

1 Introduction

1.1 About Indri

Indri is a sub version of Lemur project , which is designed for information retrieval and made by C++ . It contains apis for makeing a customized full-text search engine thst cover such as stemming ,indexing ,scoring the documents and doing the searching.

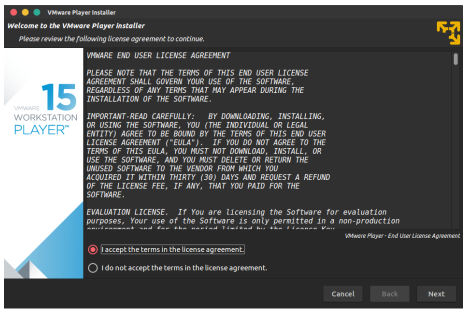
1.2 Environment Building

We prepared a Linux liked environment for the project. For a cleaning environment to testing and also contains the full GUI system , we choose to install the virtual machine in the original computer which operating system is Ubuntu Desktop 16.04 LTS.

Listing 1: Install VMware Workstation Player

```
$ wget https://www.vmware.com/go/getplayer-linux
$ sudo bash getplayer-linux
```

After the command we will get the installer of vmplayer , we just need to run install step by step .



VMware Player Installer

Welcome to the VMware Player installer

Please review the following license agreement to continue.

```
VMWARE END USER LICENSE AGREEMENT

PLEASE NOTE THAT THE TERMS OF THIS END USER LICENSE
AGREEMENT SHALL GOVERN YOUR USE OF THE SOFTWARE,
REGARDLESS OF ANY TERMS THAT MAY APPEAR DURING THE
INSTALLATION OF THE SOFTWARE.

IMPORTANT-READ CAREFULLY:   BY DOWNLOADING, INSTALLING,
OR USING THE SOFTWARE, YOU (THE INDIVIDUAL OR LEGAL
ENTITY) AGREE TO BE BOUND BY THE TERMS OF THIS END USER
LICENSE AGREEMENT ("EULA").  IF YOU DO NOT AGREE TO THE
TERMS OF THIS EULA, YOU MUST NOT DOWNLOAD, INSTALL, OR
USE THE SOFTWARE, AND YOU MUST DELETE OR RETURN THE
UNUSED SOFTWARE TO THE VENDOR FROM WHICH YOU
ACQUIRED IT WITHIN THIRTY (30) DAYS AND REQUEST A REFUND
OF THE LICENSE FEE, IF ANY, THAT YOU PAID FOR THE
SOFTWARE.

EVALUATION LICENSE.  If You are licensing the Software for evaluation
purposes, Your use of the Software is only permitted in a non-production
```

VMware Player - End User License Agreement

[✓ I accept the terms in the license agreement.]

I do not accept the terms in the license agreement.

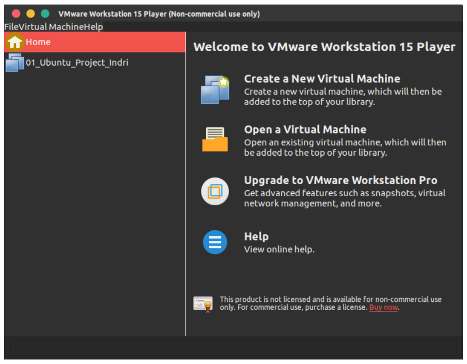
Figure 1: Intital Screenshot of vmplayer installer

Next we need the Ubuntu 16.04 ISO file for the vmplayer.

Listing 2: Get the iso file of Ubuntu 16.04

```
$ wget http://releases.ubuntu.com/16.04/ubuntu-16.04.6-desktop-amd64.iso
```

Now we just need to run the vmplayer and install the Ubuntu.We setting our testing environment called 01_Ubuntu_Project_Indri .


VMware Workstation 15 Player (Non-commercial use only)
File Virtual Machine Help
Home
01_Ubuntu_Project_Indri

Welcome to VMware Workstation 15 Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Upgrade to VMware Workstation Pro
Get advanced features such as snapshots, virtual network management, and more.

Help
View online help.

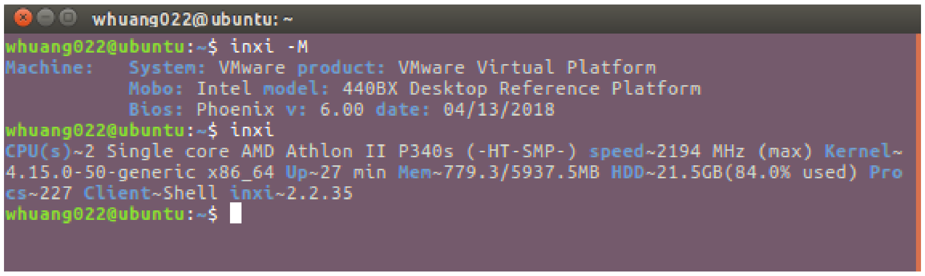
This product is not licensed and is available for non-commercial use only. For commercial use, purchase a license. Buy now.

Figure 2: Intital Screenshot of vmplayer

For checking the system , we can type :

Listing 3: Get system info

```
$ sudo apt install inxi
$ inxi -M
$ inxi
```



```
whuang022@ubuntu: ~
whuang022@ubuntu:~$ inxi -M
Machine:    System: VMware product: VMware Virtual Platform
            Mobo: Intel model: 440BX Desktop Reference Platform
            Bios: Phoenix v: 6.00 date: 04/13/2018
whuang022@ubuntu:~$ inxi
CPU(s)~2 Single core AMD Athlon II P340s (-HT-SMP-) speed~2194 MHz (max) Kernel~
4.15.0-50-generic x86_64 Up~27 min Mem~779.3/5937.5MB HDD~21.5GB(84.0% used) Pro
cs~227 Client~Shell inxi~2.2.35
whuang022@ubuntu:~$
```

Figure 3: Checking system info

1.3 Install Indri

We get Indri 5.8 :

Listing 4: Install Indri in bash

```
$ wget https://sourceforge.net/projects/lemur
/files/lemur/indri-5.8/indri-5.8.tar.gz/download -O indri-5.8.tar.gz
$ tar xvfz indri-5.8.tar.gz
```

After unzip , we can see the Indri 5.8 folder :

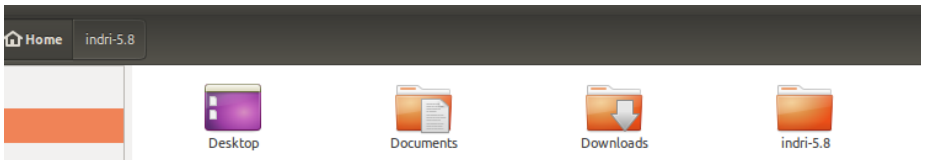


Figure 4: Checking system info

1.3.1 Installation of dependencies

There is the list [1] that we need to install before compiling the Indri.

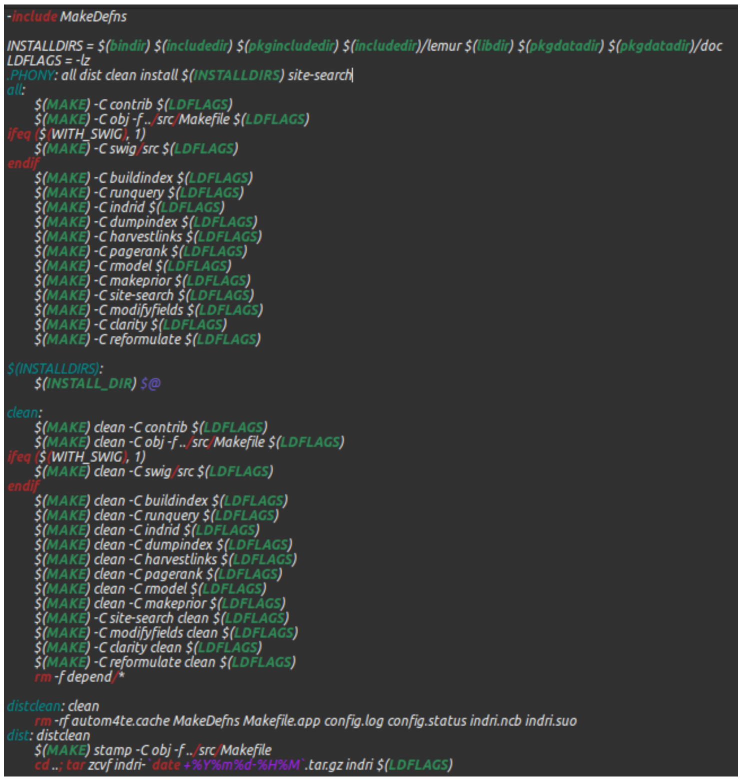
1. build-essential:the g++ stuff.
2. git : the version controll system.
3. vim : the bash editor.
4. zlibc : compressed file-system using in Indri .
5. zlib1g : compressed file-system using in Indri .
6. zlib1g-dev : compressed file-system using in Indri

So we just typing :

Listing 5: Get dependencies

```
$ apt-get install build-essential
git vim wget zlibc zlib1g zlib1g-dev -y
$ apt-get update
```

For the zlib , we need to change the Makefile [2] , and add the compiling flag " -lz " .



```
-include MakeDefns

INSTALLDIRS = $(bindir) $(includedir) $(pkgincludedir) $(includedir)/lemur $(libdir) $(pkgdatadir) $(pkgdatadir)/doc
LDFLAGS = -lz
.PHONY: all dist clean install $(INSTALLDIRS) site-search
all:
	$(MAKE) -C contrib $(LDFLAGS)
	$(MAKE) -C obj -f ../src/Makefile $(LDFLAGS)
ifeq ($(WITH_SWIG), 1)
	$(MAKE) -C swig/src $(LDFLAGS)
endif
	$(MAKE) -C buildindex $(LDFLAGS)
	$(MAKE) -C runquery $(LDFLAGS)
	$(MAKE) -C indrid $(LDFLAGS)
	$(MAKE) -C dumpindex $(LDFLAGS)
	$(MAKE) -C harvestlinks $(LDFLAGS)
	$(MAKE) -C pagerank $(LDFLAGS)
	$(MAKE) -C rmodel $(LDFLAGS)
	$(MAKE) -C makeprior $(LDFLAGS)
	$(MAKE) -C site-search $(LDFLAGS)
	$(MAKE) -C modifyfields $(LDFLAGS)
	$(MAKE) -C clarity $(LDFLAGS)
	$(MAKE) -C reformulate $(LDFLAGS)

$(INSTALLDIRS):
	$(INSTALL_DIR) $@

clean:
	$(MAKE) clean -C contrib $(LDFLAGS)
	$(MAKE) clean -C obj -f ../src/Makefile $(LDFLAGS)
ifeq ($(WITH_SWIG), 1)
	$(MAKE) clean -C swig/src $(LDFLAGS)
endif
	$(MAKE) clean -C buildindex $(LDFLAGS)
	$(MAKE) clean -C runquery $(LDFLAGS)
	$(MAKE) clean -C indrid $(LDFLAGS)
	$(MAKE) clean -C dumpindex $(LDFLAGS)
	$(MAKE) clean -C harvestlinks $(LDFLAGS)
	$(MAKE) clean -C pagerank $(LDFLAGS)
	$(MAKE) clean -C rmodel $(LDFLAGS)
	$(MAKE) clean -C makeprior $(LDFLAGS)
	$(MAKE) -C site-search clean $(LDFLAGS)
	$(MAKE) -C modifyfields clean $(LDFLAGS)
	$(MAKE) -C clarity clean $(LDFLAGS)
	$(MAKE) -C reformulate clean $(LDFLAGS)
	rm -f depend/*

distclean: clean
	rm -rf autom4te.cache MakeDefns Makefile.app config.log config.status indri.ncb indri.suo
dist: distclean
	$(MAKE) stamp -C obj -f ../src/Makefile
	cd ..; tar zcvf indri-`date +%Y%m%d-%H%M`.tar.gz indri $(LDFLAGS)
```

Figure 5: edit Makefile

1.3.2 Installation of Indri

Listing 6: Install Indri in bash

```
$ cd ~
$ cd indri-5.8
$ sudo ./configure
$ make
$ sudo make install
```

The system will have Indri after the installation.

2 Using Indri to indexing WT2G

Fist we creat the path of workspace and download the dataset WT2G.And creat the path for indexing.

Listing 7: Download WT2G File

```
$ cd ~
$ mkdir workspace
$ cd workspace
$ wget http://wm5.nccu.edu.tw/base/10001/
course/10021115/content/proj02/WT2G.tbz
$ tar jxf WT2G.tbz
$ mkdir ~/workspace/WT2G_index
$ mkdir ~/workspace/WT2G_index_no_stemming
$ mkdir ~/workspace/WT2G_index_parameter_file
$ mkdir ~/workspace/WT2G_run_shell
```

Next we write the index parameter files ,the format can find at [3] . There will be two types of index file , one is stemming by porter and the other one do not have stemming.

Listing 8: project_W2T.xml

```
<parameters>
<memory>1G</memory>
<stemmer>
  <name>porter</name>
</stemmer>
<index>/home/whuang022/workspace/WT2G_index</index>
<corpus>
  <path>/home/whuang022/workspace/WT2G</path>
  <class>trecweb</class>
</corpus>
</parameters>
```

Listing 9: project_W2T_withoutstemming.xml

```
  <parameters>
    <memory>1G</memory>
    <index>/home/whuang022/workspace/WT2G_index_no_stemming</index>
    <corpus>
        <path>/home/whuang022/workspace/WT2G</path>
        <class>trecweb</class>
    </corpus>
</parameters>
```

The index parameters files will save at path
~/workspace/WT2G_index_parameter_file

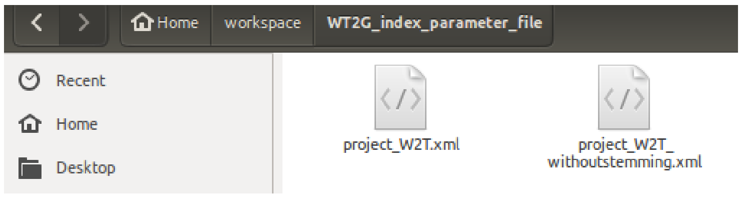


Figure 6: The index parameter files

Then we write shell script to do indexing.We put all shell script in the path ∼/workspace/WT2G_run_shell

We can use the executable file IndriBuildIndex to build index from bash command . The command format is IndriBuildIndex [*index_parameter_file_path*]

Listing 10: index.sh (use porter stemming)

```

#!/bin/bash

cd ~/indri-5.8/buildindex

if [ -e IndriBuildIndex ]
then
    echo "IndriBuildIndex [OK] "
    ./IndriBuildIndex ~/workspace/WT2G_index_parameter_file
    /project_W2T.xml
else
    echo "IndriBuildIndex [Error] "
fi
```

Listing 11: index_no_stemming.sh (no stemming)

```

#!/bin/bash

cd ~/indri-5.8/buildindex

if [ -e IndriBuildIndex ]
then
    echo "IndriBuildIndex [OK] "
    ./IndriBuildIndex ~/workspace/WT2G_index_parameter_file
    /project_W2T_withoutstemming.xml
else
    echo "IndriBuildIndex [Error] "
fi
```

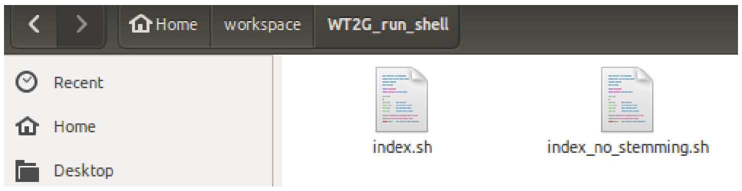


Figure 7: The index shell scripts

Finally , we just run the shell scripts , ande Indri will generate the index files.

Listing 12: Run index shell scripts

```
$ cd ~/workspace/WT2G_run_shell
$ sudo bash index.sh
$ sudo bash index_no_stemming.sh
```

The results is in the index path.

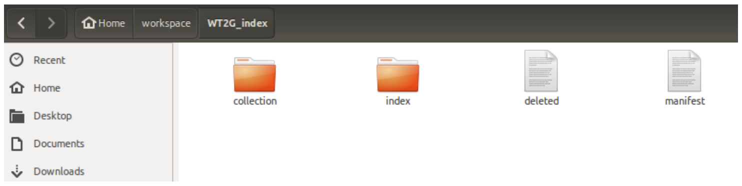


Figure 8: The index files generate from IndriBuildIndexs

3 Retrieval Files form Indri index files

3.1 Query file and scoring functions

We can use the executable file IndriRunQuery to doing the retrieval from command ,the scoring can write in the query xml file.The detail information of the format about query file , can read at [4]. The important parameters are:

1. <index>:the path of index files
2. <count> : the number of retun files
3. <baseline> : the basic scoring function like Okapi TFIDF
4. <method>: the other scoring functions
5. <query>: the query

3.1.1 Vector space model Okapi TFIDF

From the project document there is :

> You will have to use for the weights OKAPI TF x IDF where $OKAPITF = tf/(tf + 0.5 + 1.5 * doclen/avgdoclen)$. For queries, Okapi TF can also be computed in the same way, just use the length of the query to replace doclen. Also note that the definition of OKAPI TF is$tf/tf + k1((1-b)+b*doclen/avgdoclen)$. In the above formula, you can set k1 = 2 and b = 0.75, to end up with:$tf/(tf + 0.5 + 1.5 * doclen/avgdoclen)$.

So we just need to set okapi with k1=2.0, b:0.75. The query file is :

Listing 13: okapi.xml

```xml
  <parameters>
    <baseline>okapi,k1=2.0, b:0.75</baseline>
    <index>/home/whuang022/workspace/WT2G_index</index>
    <count>1000</count>
    <query>
        <type>indri</type>
        <number>401</number>
        <text>
          foreign minorities germany
        </text>
    </query>
</parameters>
```

3.1.2 Language modeling with Laplace Smoothing

Because Indri dose not have an implement of laplace smoothing , so we need to implement the customized scoring function .In Indri project the scoring functions has an abstract base class named TermScoreFunction.hpp which was designed for define the scoring function class‘s api.

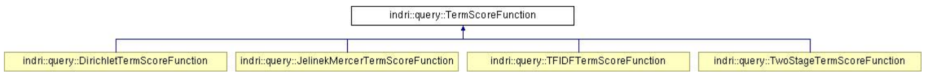

indri::query::TermScoreFunction
indri::query::DirichletTermScoreFunction
indri::query::JelinekMercerTermScoreFunction
indri::query::TFIDFTermScoreFunction
indri::query::TwoStageTermScoreFunction

Figure 9: indri::query::TermScoreFunction Class Reference [5]

Listing 14: TermScoreFunction.hpp

```
#ifndef INDRI_TERMSCOREFUNCTION_HPP
#define INDRI_TERMSCOREFUNCTION_HPP
namespace indri
{
  namespace query
  {

    class TermScoreFunction {
    public:
      virtual double scoreOccurrence( double occurrences, int contextLength ) = 0;
      virtual double scoreOccurrence( double occurrences, int contextLength, double documentOccurrences, int documentLength ) = 0;
    };
  }
}
#endif // INDRI_TERMSCOREFUNCTION_HPP
```

For the customized scoring function , the basic design for the class that extends TermScoreFunction is implement the first function ”scoreOccurrence(double occurrences, int contextLength)” , and return the first function‘s result in the second function. The laplace smoothing mathod is as the following :$p_i = \frac{m_i+1}{n+k}$where m = term frequency, n=number of terms in document (doc length) , k=number of unique terms in corpus. In this implementation , we use log transform for p_i to $log\,(p_i)$. The ”occurrences” is term frequency , the ”contextLength” is number of terms in document.However,we need the size of unique terms in corpus. So we need to use QueryEnvironment.hpp and it‘s api termCountUnique () to get k.The following is the code :

Listing 15: LaplaceTermScoreFunction.hpp

```

#ifndef INDRI_LAPLACETERMSCOREFUNCTION_HPP
#define INDRI_LAPLACETERMSCOREFUNCTION_HPP
#include <indri/QueryEnvironment.hpp>
#include <math.h>
#include <iostream>
namespace indri
{
  namespace query
  {
    class LaplaceTermScoreFunction : public TermScoreFunction {
    private:
      double _alpha;
      indri::api::QueryEnvironment *env;
    public:
      LaplaceTermScoreFunction(double alpha , const std::string& indexName) {
        _alpha = alpha;
          env = new indri::api::QueryEnvironment();
          env->addIndex(indexName);
      }
      double scoreOccurrence( double occurrences, int contextSize ) {
        double seen = ( double(occurrences) + _alpha ) / ( double(contextSize) + double(env->termCountUnique ()) );
        return log( seen );
      }
      double scoreOccurrence( double occurrences, int contextSize, double documentOccurrences, int documentLength ) {
          return scoreOccurrence( occurrences, contextSize );
      }
    };
  }
}
```

```

#endif // INDRI_LAPLACETERMSCOREFUNCTION_HPP
```

To add this customized class to Indri wheile calling the IndriRunQuery , we need to add the mapping function in src/TermScoreFunctionFactory.cpp .We add the rule to calling our laplace class :

Listing 16: TermScoreFunctionFactory.cpp

```

... (some other codes )
else if ( method == "laplace" || method == "l" || method == "lap" ){
    // laplace -- takes parameter "alpha" & index path
    double alpha = spec.get( "alpha", 1.0 );
    return new indri::query::LaplaceTermScoreFunction(alpha,spec.get( "index", "" ));
  }
... (some other codes )
```

Finally , we need to re compile the code (use "make" and "sudo make install") , the new executable file will contain the customized api design by us ,and the query file is :

Listing 17: laplace.xml

```
  <parameters>
    <method>laplace,alpha=1.0,index=/home/whuang022/workspace/WT2G_index</method>
    <index>/home/whuang022/workspace/WT2G_index</index>
    <count>1000</count>
    <query>
        <type>indri</type>
        <number>401</number>
        <text>
          foreign minorities germany
        </text>
    </query>
</parameters>
```

3.1.3 Jelinek-Mercer smoothing using the corpus

The formula for Jelinek-Mercer smoothing is : $p_i = \lambda P + (1 - \lambda)\, Q$ and we select $\lambda = 0.8$.

Therefore , the query file is :

Listing 18: jelinek_mercer.xml

```
  <parameters>
    <rule>method:linear,collectionLambda:0.8,documentLambda:0.2</rule>
    <index>/home/whuang022/workspace/WT2G_index</index>
    <count>1000</count>
    <query>
        <type>indri</type>
        <number>401</number>
        <text>
          foreign minorities germany
        </text>
    </query>
</parameters>
```

3.1.4 Jelinek-Mercer smoothing add Laplace

Listing 19: JelinekMercerTermLaplaceScoreFunction.hpp

```



//
// JelinekMercerTermLaplaceScoreFunction
#ifndef INDRI_JELINEKMERCERLAPLACETERMSCOREFUNCTION_HPP
#define INDRI_JELINEKMERCERLAPLACETERMSCOREFUNCTION_HPP
#include <indri/QueryEnvironment.hpp>
#include <math.h>
namespace indri
{
  /// indri query processing and scoring components.
  namespace query
  {

    class JelinekMercerLaplaceTermScoreFunction : public TermScoreFunction {
    private:
      double _lambda;
      double _backgroundLambda;
      double _collectionFrequency;
      double _collectionComponent;
      double _oneLevelCollectionComponent;
      double _contextLambda;
      double _collectionLambda;
      double _documentLambda;
      double _foregroundLambda;
      indri::api::QueryEnvironment *env;
    public:
      JelinekMercerLaplaceTermScoreFunction( double collectionFrequency, double collectionLambda, double documentLambda
, const std::string& indexName ) {
        _contextLambda = (1 - collectionLambda - documentLambda);
        _collectionFrequency = collectionFrequency;
        _collectionLambda = collectionLambda;
        _documentLambda = documentLambda;
        _foregroundLambda = (1 - _collectionLambda);
        env = new indri::api::QueryEnvironment();
  env->addIndex(indexName);

        assert( _documentLambda >= 0.0 && _documentLambda <= 1.0 );
        assert( _collectionLambda >= 0.0 && _collectionLambda <= 1.0 );
        assert( _contextLambda >= 0.0 && _contextLambda <= 1.0 );

        _collectionComponent = _collectionLambda * _collectionFrequency;
      }

      double scoreOccurrence( double occurrences, int contextSize ) {
        double seen2 = ( double(occurrences) +1.0 ) / ( double(contextSize) + double(env->termCountUnique ()) );
        double contextFrequency = contextSize ? occurrences / double(contextSize) : 0.0;
        return (log( _foregroundLambda * contextFrequency + _collectionComponent )+log(seen2));
      }

      double scoreOccurrence( double occurrences, int contextSize, double documentOccurrences, int documentLength ) {
        double seen2 = ( double(occurrences) +1.0 ) / ( double(contextSize) + double(env->termCountUnique ()) );
        double contextFrequency = contextSize ? occurrences / double(contextSize) : 0.0;
        double documentFrequency = documentLength ? documentOccurrences / double(documentLength) : 0.0;
        return (log( _contextLambda * contextFrequency + _documentLambda * documentFrequency + _collectionComponent )+log(seen2));
      }
    };
  }
}

#endif // INDRI_JELINEKMERCERLAPLACETERMSCOREFUNCTION_HPP
```

3.2 The shell script to Retreival

For example of laplace we will have :

Listing 20: Run index shell scripts

```
#!/bin/bash

cd ~/indri-5.8/runquery

if [ -e IndriRunQuery ]
then
     echo "IndriRunQuery [OK] "
     ./IndriRunQuery ~/workspace/WT2G_query_parameter_file/laplace.xml -trecFormat=true > ~/workspace/WT2G_result/output_laplace.txt
     cd ~/workspace/WT2G_result/
     perl trec_eval.pl [-q] qrels.401-450.txt output_laplace.txt > laplace_evaluation.txt
else
     echo "IndriRunQuery [Error] "
fi
```

4 Result

4.1 Un-interpolated Mean Average Precision

	Okapi TF	Laplace	Jelinek Mercer	Jelinek Mercer add Laplace
stemming	0.2704	0.5123	0.0591	0.5123
no stemming	0.1932	0.3205	0.0407	0.3205

4.2 Precision at rank 10 documents

	Okapi TF	Laplace	Jelinek Mercer	Jelinek Mercer add Laplace
stemming	0.3000	0.9000	0.1000	0.9000
no stemming	0.3000	0.8000	0.0000	0.8000

In conclution , if we do not stemming the performace is not good at all.

4.3 Access the project

Listing 21: Access the project

```

 1. The source code with customized class :
https://github.com/whuang022nccu/IndriLab
2. The shell scripts (.sh) , parameter files (.xml) ,and reslut files :
https://github.com/whuang022nccu/IndriLab/tree/master/parameter%20files%20
```

References

[1] Can't Make - error, https://sourceforge.net/p/lemur/discussion/2106523/thread/3abe18beed/, 25 04 2019.

[2] Compilation problems with ZLIB, https://stackoverflow.com/questions/9700414/compilation-problems-with-zlib/18875275#18875275, 16 03 2012.

[3] IndriBuildIndex Parameters, David Fisher , https://sourceforge.net/p/lemur/wiki/IndriBuildIndex%20Parameters/

[4] Specifying Retrieval Parameters, David Fisher , https://sourceforge.net/p/lemur/wiki/IndriRunQuery/

[5] indri::query::TermScoreFunction Class Reference , https://lemur.sourceforge.io/indri/classindri_1_1query_1_1TermScoreFunction.html