Programming HW1

Web Retrieval and Mining Spring 2020

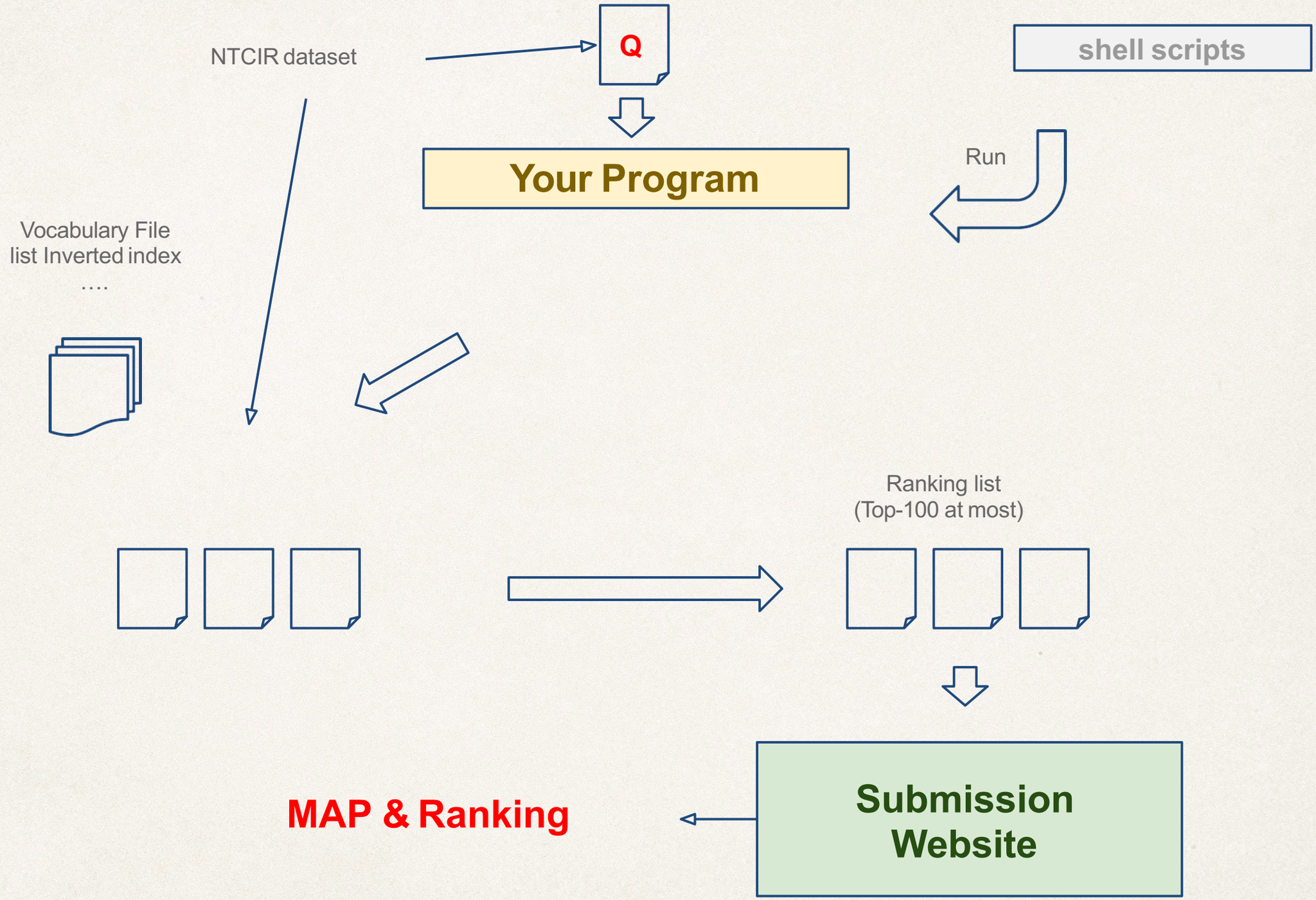
2020/4/3

Introduction

- In this homework, you are asked to implement **a small information retrieval system**.

- We will give you a bunch of Chinese news articles and several queries in NTCIR format, and your task is to find the relevant documents among these articles according to the given queries.

- You should implement the retrieval system by **Vector Space Model** (VSM) with **Rocchio Relevance Feedback** (pseudo version).


NTCIR dataset
Q
shell scripts
Your Program
Run
Vocabulary File
list Inverted index
....
Ranking list
(Top-100 at most)
MAP & Ranking
Submission
Website

NTCIR Document Set

- Please sign up the **USER AGREEMENT FORM** and hand it to the TAs (at R302 of CSIE dept.) in order to use this corpus.
  - Note that you'll get **NO POINTs** if you don't sign up the user agreement form.
    - **DO NOT** distribute this dataset
- Download NTCIR document set on kaggle.
- We have indexed the NTCIR documents and produced three MODEL FILES for you:
  - vocab.all
  - file-list
  - inverted-file

NTCIR Document Format

- The NTCIR document format conforms to XML 1.0, and make use a limited set of tags to represent different semantic levels of newswire texts.
- The root element is <xml>, it contains only one <doc> tag.
- A <doc> tag represents exactly one newswire article, in which several sub-elements are used to specify different type of information:
  - <id>: An unique document ID.
  - <date>: The publication date.
  - <title>: The title of the article.
  - <text>: The content of the article, which may include one or more passages enclosed in <p> tags.

Format: *vocab.all*

```
utf8
Valentine
Powell
Copper
version
EGCG
Powerb
Powers
...
```

- This file contains all vocabularies in NTCIR documents.
- The first line is character encoding format.
- Each line of the following is a vocabulary.
- Vocabularies are case-sensitive.
- Each vocabulary will have a *vocab_id* according to its line number.
  - E.g. the *vocab_id* of "Valentine" is 1; "Powell" is 2.

Format: *file-list*

```
./CIRB010/cdn/chi/cdn_chi_0000001
./CIRB010/cdn/chi/cdn_chi_0000002
./CIRB010/cdn/chi/cdn_chi_0000003
./CIRB010/cdn/chi/cdn_chi_0000004
./CIRB010/cdn/chi/cdn_chi_0000005
...
```

- This is a list of all NTCIR documents.
- Each line denotes a document which has its line number (start from 0) as its file_id.
  - E.g.
  - ./CIRB010/cdn/chi/cdn_chi_0000001 has file_id 0,
  - ./CIRB010/cdn/chi/cdn_chi_0000002 has file_id 1.

Format: *inverted-file*

```
1 -1 2
33689 1
38365 1
2 -1 1
33256 1
```

- *vocab_id* and *file_id* referred from *vocab.all* and *file-list*.

- *vocab_id_1 vocab_id_2* denotes an **unigram** when *vocab_id_2*==-1 or a **bigram** when *vocab_id_2*!=-1.

- If there are **N** files containing *vocab_id_1 vocab_id_2*, there will be the number **N** next to vocab_id_2, followed by **N lines** that display the counts of this term in each file.

Program IO

- Your program is required to support input of a **query file**, and output a **ranking list**. (Please see Query File Format and Ranking List Format next pages)

- We provide 30 query topics for you as inputs. (only 10 with answers and the others are used to evaluate your performance)

- There is no restriction to the programming language you use, but make sure your program is **executable on R217 workstation.**

- Using the third party tools directly for VSM or Relevance Feedback is prohibited.

Query File Format

- The NTCIR topic format conforms to XML 1.0, in which the document is rooted at an <xml> tag.
- The file contains multiple topics, each of them is enclosed in a <topic> tag. In each topic, different types of information are specified by the following tags:
  - <number>: The topic number.
  - <title>: The topic title.
  - <question>: A short description about the query topic.
  - <narrative>: Even more verbose descriptions about the topic.
  - <concepts>: A set of keywords that can be used in retrieval about the topic.
- You have to retrieve several relevant documents for each topic.
- All the content of title, question, narrative, and concepts can be used as the query of the topic, it's your own choice to decide which part(s) you want to use.

Ranking List Format

- The first line includes two column names: "*query_id*", "*retrieved_docs*"
- First column: ***query_id,*** which is the **last three digits** in <number>…</number> tag in the query xml file.
- Second column: ***document_ids,*** which is the string in <id>…</id> tag in the NTCIR document. Please note it should be in lowercase.
- The two columns should be separated by a comma.
- Document ids should be separated by spaces**.**
- Note that retrieved docs should be sorted by their ranks
- You can retrieve **at most 100 documents** for each topic.

Program Execution Details

- You are given two shell scripts to compile and run your program.
- You should edit these two scripts according to how you implement this assignment.
- When testing your program, we will execute the following commands on **R217 workstation**, please make sure your program is executable on the workstation.
  - $./compile.sh
  - $./execute.sh -option1 value1 -option2 value2...

Program Execution Details (con't)

- Here are the required options that must be supported by your program. (Options without default values are guaranteed to be specified when we test your program.)

```
SYNOPSIS:
  execute.sh [-r] -i query-file -o ranked-list -m model-dir -d NTCIR-dir

OPTIONS:
  -r
      If specified, turn on the relevance feedback in your program.
  -i query-file
      The input query file.
  -o ranked-list
      The output ranked list file.
  -m model-dir
      The input model directory, which includes three files:
          model-dir/vocab.all
          model-dir/file-list
          model-dir/inverted-index
  -d NTCIR-dir
      The directory of NTCIR documents, which is the path name of CIRB010 directory.
      ex. If the directory's pathname is /tmp2/CIRB010, it will be "-d /tmp2/CIRB010".
```

Restrictions

- You should generate features like tf-idf ,implement **VSM** and **Rocchio Relevance Feedback** by yourself without using any other packages.

- If you are not sure packages you used is legal or not, please inquiry TA by e–mail.

- Your program should finish in 5 minutes.

- Do not copy other's code. Those who copy code and those who allow others to copy his/her code will be punished seriously.

Evaluation

- We will use the **Mean Average Precision (MAP)** value to evaluate your ranking list.
- We provide an answer ranking list for *query-train.xml*.
  - There're two columns in the answer list, first is the *query_id*, followed by *retrieved_docs* relevant to this topic.
  - You can use this answer list to check your system's performance.
- Please produce a ranking list of *query-test.xml* and submit to Kaggle. You can see your performance ranking on the leaderboard.

Report

- Please write your report as a **Report.pdf** and put it into the zipped file. The report should contain the following content:
  - Describe your VSM (e.g., parameters….)
  - Describe your Rocchio Relevance Feedback (e.g., how do you define relevant documents, parameters…)
  - **Results of Experiments**
    - **MAP value under different parameters of VSM**
    - **Feedback vs. no Feedback**
    - Other experiments you tried
  - Discussion: what you learn in the homework.

Submission

- Please put report, scripts and code into the directory named your **student ID**. Package this folder into a zip file and submit it to NTU COOL, following is the structure and content of the zip:
- For example: R07922XXX.zip
  - +---R07922XXX(directory) (with **R** in uppercase)
    - +---Report.pdf
    - +---compile.sh
    - +---execute.sh
    - +---All the other files and source code required by your program
    - (Note that you don't need to submit the model files and NTCIR documents)

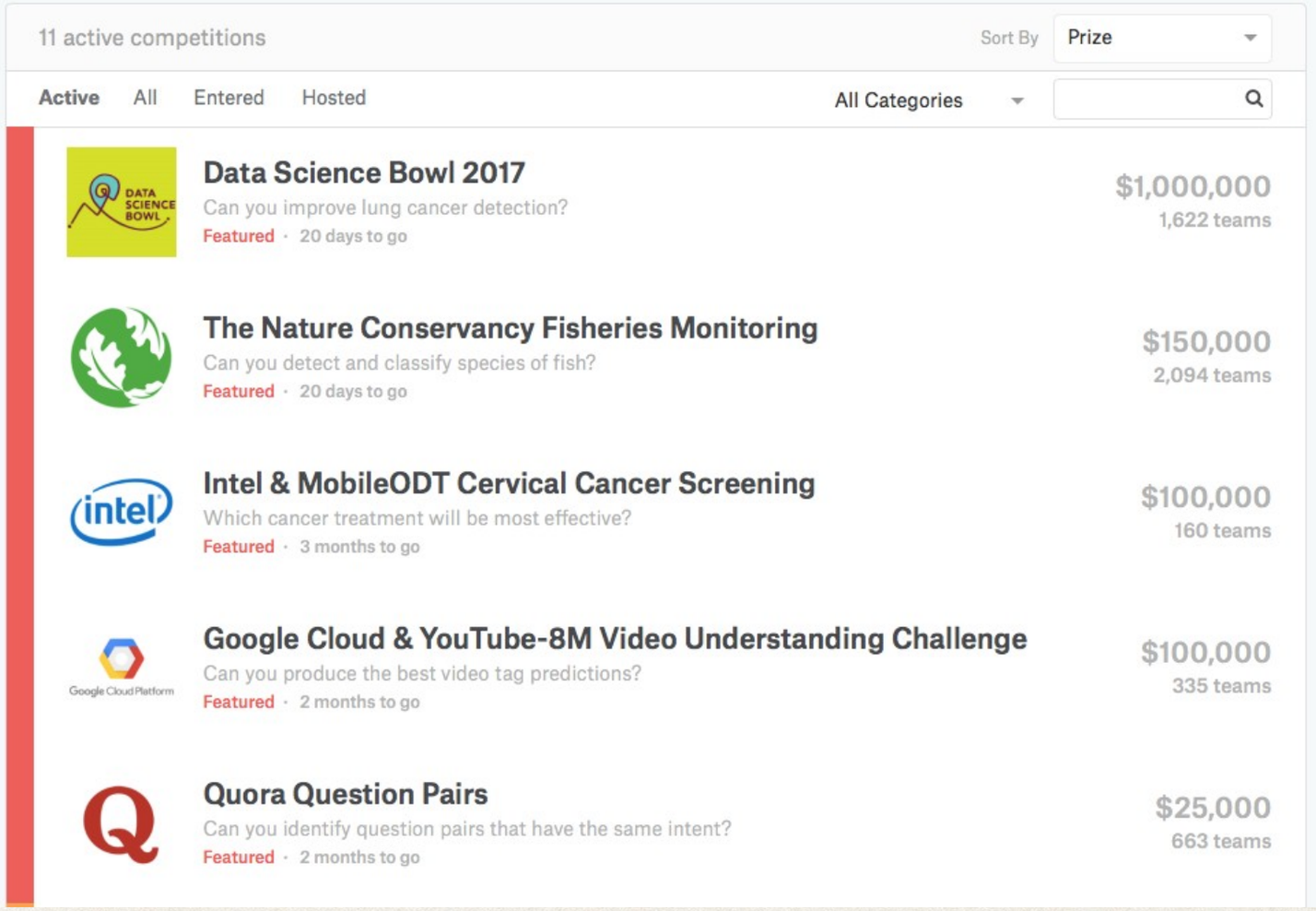
Scoring

- 20% for VSM model.
- 10% for Rocchio relevance feedback.
- 20% for your report.
- 25% for performance better than simple baseline on public leaderboard
- 25% for performance better than strong baseline on public leaderboard
- Note that you'll get 0 for performance if you don’t have record on the ranking website
- Note that you'll get 0 if you don't sign up the user agreement form.

Competition on kaggle

kaggle

- https://www.kaggle.com



Join Competition

- This is individual homework. One person in each team.

- The link of the competition is below:

- https://www.kaggle.com/c/wm-2020-vsm-model/

kaggle

Search

Home

Compete

Data

Notebooks

Discuss

Courses

More

Recently Viewed

WM 2017 - VSM Model

InClass Prediction Competition

WM 2020 - VSM Model

build a document retrieval system!

Host | Overview | Data | Notebooks | Leaderboard | Rules | Team | My Submissions

This competition hasn't been launched. Only hosts and Kaggle admins can see it.

Overview

Edit

Description

Evaluation

+ Add Page

We will give you a bunch of **Chinese news articles** and several queries in **NTCIR** format, and your task is to find the relevant documents among these articles according to the given queries.

You should implement the retrieval system by **Vector Space Model (VSM)** with **Rocchio Relevance Feedback (pseudo version)**.

The timeline for this competition will appear once there is a valid start date and deadline.

Points **This competition does not award standard ranking points**

Tiers **This competition does not count towards tiers**

Bonus

- Extra score for top-10 ranking on public and private leaderboard respectively
  - 3% for 1st -3rd
  - 2% for 4th -5th
  - 1% for 6th -10th
- rank 1 at public, rank 5 at private ➡ 5 points

Leaderboard

- Public/Private leaderboard
- 10/10 queries for public and private respectively
- Best on public ≠ best on private

Rules

- One account per participant
- The name on the leaderboard must be your student ID(with upper case).
- You may select up to 2 final submissions for judging.
- You may submit a maximum of 5 entries per day.

Deadline

- Deadline: 2020/04/19 23:59:59 (UTC+8)
- Late policy: 10% per day
- Or email to TAs: irlab.ntu@gmail.com