CS4740/5740 Introduction to NLP
Fall 2016
Question Answering

Part 1: due via CMS by Wednesday, 11/02 11:59pm
Part 2: due via CMS by Sunday, 11/13 11:59pm

1 Overview

Goal for this project: To gain a bit of experience in the design, implementation, and evaluation of question-answering (QA) systems. To gain experience in working with standard off-the-shelf NLP components.

As in previous assignments, this assignment is fairly open-ended. You are to implement a QA system that will operate in one of the standard TREC QA frameworks: the input to the system is a question, the output is a ranked list of five guesses for the answer. No human intervention is allowed in deriving answers. For each part below, assume that your system has entered the 50-byte (short answer) QA contest (i.e., "track" in TREC terminology) so **all answers should be 10 or fewer words in length**. The main metric for evaluation is the **mean reciprocal rank**[1]. To simplify the task, the type of a question in this assignment can only be ***who***, ***where***, and ***when***.

2 Grouping

We recommend groups of 3-4 people.

3 Project Phases

3.1 Part 1

3.1.1 Introduction

For Part 1, we provide a QA *development* corpus that contains a set of questions (in total 232 questions, with ID from 89 to 320) and the expected answer(s) for each question. Since we cannot make available to you the actual 9GB TREC collection used in the TREC QA studies, we instead provide the top 100 documents retrieved by the Smart IR system for each question in the corpus. Answers to

[1] https://en.wikipedia.org/wiki/Mean_reciprocal_rank

each question are to be extracted from these 100 documents. Note that it is possible for some questions that none of the 100 retrieved documents contains the answer.

In Part 1, you are supposed to:

- Design and implement a baseline system for this project.
- Run, evaluate and analyze your baseline on the development corpus.
- Work out a proposal for your final system.

3.1.2 What We Provide

We provide you with the development corpus including:

- **question.txt:** the questions.
- **pattern.txt:** expected answer(s) found by TREC assessors for each question. Each line consists of the question ID followed by a regular expression which corresponds to possible answers.
- **eval.pl:** a scoring program that can be used to score your answer file.
  - To use this program, you need to install Perl[2].
  - To score your answers, simply execute the following command:

    ```
    perl eval.pl pattern.txt answer.txt
    ```

    where answer.txt is your answer file whose format is described below.
  - The answer file should contain the following for each question:

    question_id document_id answer-text(for top-ranked guess)
    question_id document_id answer-text(for second guess)
    question_id document_id answer-text(for third guess)
    question_id document_id answer-text(for fourth guess)
    question_id document_id answer-text(for fifth guess)

    The *document_id* refers to the file name of the document where the answer string was found. Use "nil" as the answer-text if your system finds no answer for a particular question. The answer-text should be 10 or fewer words.
- **Top 100 documents retrieved for each question:** the top 100 documents retrieved for each question by Smart. The folder name corresponds to the question ID.

[2] https://www.perl.org/

3.2 Part 2

3.2.1 Introduction

Your goal for Part 2 is to:

- Implement your final QA system based on your proposal in Part 1.
- Run, evaluate and analyze your final system on the development corpus.
- Run your final system on the test corpus which will be available **one week** before the due date of the assignment.

Note that the idea for the test corpus is to run your system on these questions **a single time without modifying your final system or your answer file**. We will evaluate your answer file and tell you the result.

3.2.2 What We Will Provide

We will provide you with the test corpus including:

- **question.txt:** a set of questions in the same format as Part 1 (in total 88 questions, with ID from 1 to 88). You will not be given the expected answers to them.
- **Top 100 documents retrieved for each question**.

4 What to Turn in

4.1 Part 1 (via CMS)

- **Code of your baseline system**.
- **Answer file** produced by your baseline system for the questions from the development corpus.
- **Report** (maximum 2 pages).

4.2 Part 2 (via CMS)

- **Code of your system**.
- **Answer files** produced by your system for the questions from both the development and the test corpus.
- **Report** (maximum 8 pages).

Refer to the rubric section for detailed requirements.

5 Rubric

Note that performance does NOT play an important factor in this assignment. We will mainly take into account your design, implementation and analysis of your system.

5.1 Part 1 (20 pts)

- (8 pts) **Code and Output**
  - (4 pts) If you can show some progress of building your baseline system.
  - (4 pts) If you system can generate the answer file in correct format.
- (12 pts) **Report**
  - (2 pts) Metadata: write names and netIDs of every member in your group on the first page of your report.
  - (6 pts) Report of your baseline system
    - (2 pts) A description of your baseline system (or your plan of building your baseline system if it has not been finished).
    - (2 pts) A short justification for the baseline system of your choice.
    - (2 pts) Result analysis for your system on some portion of the development corpus.
  - (4 pts) A proposal for your final system. This could include a series of improvements over your baseline system or could be a completely different approach. This proposal is your opportunity to get feedback on your final system, so please make use of it.
  - (0 pt) You must include a section describing how each member contributed to the project.

5.2 Part 2 (80 pts)

- (22 pts) **Code and Output**
  - (2 pts) Only include code that you wrote your selves. DO NOT include code from existing toolkits. DO NOT include data we have.
  - (10 pts) The output file of answers produced by your system for the questions from the **development corpus** that we provided.
    - (2 pts) It is submitted.
    - (8 pts) It is in correct format.
  - (10 pts) The output file of answers produced by your system for the questions from the **test corpus** that we provided.
    - (2 pts) It is submitted.

    - (8 pts) It is in correct format.

- (58 pts) **Report**
  - (2 pts) Metadata: write names and netIDs of every member in your group on the first page of your report.
  - (12 pts) A description of **your overall QA approach**. This effectively the statement of your hypothesis, so you should include justification of why you think that the particular approach will be effective.
  - (12 pts) A description of **your QA system and any baseline approaches** that you compare. Enough detail should be provided so that, in theory at least, we could re-implement it. The description should explain each component in your QA system, the steps that your system takes to answer a question, any additional online sources of information used by the system, etc. Make clear which components of the system you built yourself vs. downloaded from elsewhere vs. got from another student in the course.
  - (8 pts) A detailed walk-through of what your system did to handle one question (any one) in the corpus.
  - (24 pts) Analysis
    - (12 pts) An evaluation (e.g. using the mean reciprocal rank evaluation measure) and analysis of your system's performance on the questions from the development corpus, including a comparison with the baseline system(s).
    - (12 pts) Answer the following questions: How well did the system work? What worked? What didn't work? Can you say anything about which component is strongest/weakest?
  - (0 pt) You must include a section describing how each member contributed to the project.

6 Suggestions for How to Proceed

- Start simple! Select some really really dumb strategy to produce answers for each question just to make sure that you will have something to evaluate and to turn in. Only after you can do that should you proceed to something more sophisticated. The simple system can be your baseline against which you can compare and submit for Part 1.

- It is fine to try a strategy very different from anything discussed in class. It is even fine if the system that you produce does terribly in terms of performance. You just need to be able to argue (in your write-up) why the strategy that you investigated MIGHT have worked.

- One possibility is to try using an IR system to implement a passage retrieval strategy for question answering, i.e., a method to find the paragraph, sentence, or text snippet where the answer is likely to reside. (One option for an IR system is the Lemur system[3].)
- Another option is to instead focus on the type of question and develop a strategy specifically for that question type.

[3]Available at http://www.lemurproject.org/