Project Description and Rules

David R. Mortensen and Alan Black

February 18, 2019

A central part of 11-411/11-611 Natural Language Processing is a group project on question generation and question answering. This project is identical for both undergraduate and graduate students and undergraduate students and graduate students are free to form groups together. This project is partially judged on its intrinsic merits and partially judged based on a competition with other groups. The purpose of this document is to describe the project and the ground rules for evaluation. We will assume that you have read (if not memorized) this document.

1 Groups

Your group should have four members. Under exceptional circumstances, a group may have three or five members, but only with approval from the instructors. They may be drawn from both 11-411 and 11-611 (undergraduate and graduate) and may be drawn from either section if more than one section of the course is being offered. Groups must be formed by the end of the first week of class.

If members of your group are planning to drop the course, please let us know. It may be necessary to merge groups since it is very difficult to complete the project with fewer than three team members.

2 Tasks

2.1 Question generation

Given a document (for example, the text[1] of a document from Wikipedia), generate a set of questions relevant to that document. For example, if the document contains the sentence:

> Pittsburgh was named in 1758 by General John Forbes, in honor of British statesman William Pitt, 1st Earl of Chatham.

Your question generate system might generate the questions:

[1] Plain text, with no HTML markup

1. When was Pittsburgh named by General John Forbes, in honor of British statesman William Pitt, 1st Earl of Chatham?
2. What was named in 1758 by General John Forbes?
3. Who named Pittsburgh in 1758?

Your goal is to generate questions that are **fluent** and **reasonable**. That is, they should be in grammatical and idiomatic English and they should be answerable based on the information in the document.

2.2 Question answering

Give a plain test document and a set of questions, return an answer for each question. For example, if a document contains the following sentence:

> Pittsburgh was named in 1758 by General John Forbes, in honor of British statesman William Pitt, 1st Earl of Chatham.

And you your system is presented with the questions:

1. When was Pittsburgh named by General John Forbes, in honor of British statesman William Pitt, 1st Earl of Chatham?
2. What was named in 1758 by General John Forbes?
3. Who named Pittsburgh in 1758?
4. In honor of whom was Pittsburgh named in 1758 by General John Forbes?

It should return answers similar to:

1. 1758
2. Pittsburgh
3. General John Forbes
4. William Pitt

Your goal is to produce answers that are **fluent**, **correct**, and **concise**. That is to say, they should be grammatical/idiomatic English, they should be correct (in accord with the document), and they should not contain extraneous words.

3 Deliverables

3.1 Reports

The lion's share of your project grade will be based on your reports. These should detail what you plan to do and what you have done.

3.1.1 Initial plan

Your initial plan should be 1–2 pages in length and should answer, at a high level, the following questions:

- How are you going to use the development data to improve your system?
- Is there going to be a relationship between the asking and answering components of your system, or are you going to implement them independently?
- What tools are you going to use?
- How are you going to share code, data inside your team?
- How are you going to coordinate development inside the team?
- What technical approaches are you going to take?

3.1.2 Progress report

Your progress report should be a YouTube video (up to two minutes long) detailing what you have accomplished so far and what you are planning for the future. Any changes from your initial plan should be noted.

3.1.3 Progress report meeting

After you submit the progress report, you will meet with one instructor and one TA. They will ask you questions about your report and about your progress generally and will try to answer any questions that you have about the project and your implementation of the tasks.

3.1.4 Final report

The final report is worth almost half of the total grade. It is a YouTube video, the this time you have 6 minutes to make your case for why your system is the most clever and most innovative question generation and question answering system of all time. We value solutions that go beyond the textbook exposition of question answering and explore new avenues, even if these avenues do not perform as well as naïve systems on the competitive evaluation.

3.2 Programs

The final project submissions will be made by uploading your systems to our server. You will be provided with a group username and password about one week before the dry run. You must deliver two programs called `ask` and `answer` with the interfaces and properties listed below:

- The programs must have *exactly* the names listed above.

- The programs must be executable. If you do not know how to make this so, see Appendix A.
- The program must take exactly the command line arguments detailed below.

Both `ask` and `answer` should expect input containing non-ASCII characters in UTF-8 encoding. **Failure to address this will result in your team receiving no credit for the competitive portion of the project grade.** Debugging information should not be directed to STDOUT[2]. This will throw off the alignment of your outputs and will result in a dramatically reduced score.

3.2.1 Question generation

The `ask` program must have the following command-line interface:

```
./ask article.txt nquestions
```

Where *article.txt* is a *path* to an arbitrary plain text file (the document) and `nquestions` is an integer (the number of questions to be generated). The program should output to STDOUT a sequence of questions with each question terminated by a newline character. **Each line should not contain any text other than the question.** If your program cannot generate the requested number of questions, it **should not** try to buffer the output with empty lines.

3.2.2 Question answering

The `answer` program must have the following command-line interface:

```
./answer article.txt questions.txt
```

Where `article.txt` is a *path* to an arbitrary plain text file (the document) and `questions.txt` is a *path* to an arbitrary file of questions (one question per line with no extraneous material). The output (to STDOUT) should contain a sequence of answers with each answer terminated by a newline character. If your system cannot generate an answer for a given question, it should output a blank line or a default answer. **Each line should not contain any text other than the hypothesized answer.** Otherwise, your outputs will be penalized for conciseness.

4 Libraries and Tools

In general, you are free to use open source libraries and tools that can be installed server-side. Common examples include *NLTK*, *Stanford CoreNLP*, *Berkeley Parser*, and *spaCy*. One exception involves grammar checking libraries, which you may use only if you implement them yourself. **You may use web services**

[2]Instead, use a logging library or write you messages to STDERR

only under exceptional circumstances and only with the approval of the instructors. If you project is found to access an external service without permission, you will receive no credit.

5 Evaluation and Grading

The project grades will be calculated as follows:

Deadline	What's Graded	Value
5 Mar	Initial plan	2 points
5 Mar	Progress report	3 points
—TBA—	Instructor meeting	3 points
9 Apr	Dry run system	2 points
25 Apr	Final working system	6 points
2 May	Final project report	14 points
2 May	Awards	

Project reports and meetings will be graded according to the following criteria:

- **Soundness.** Are the theoretical and empirical foundations of the project, as reflected in the report, strong. (50%)
- **Novelty.** Do the ideas presented in the report show originality or creativity? How much do they go beyond the templates given in the textbook and in lecture? (25%)
- **Clarity.** Is the material presented in a clear and compelling way? (25%)

The dry run system will be graded according to the following criteria:

- `ask` runs on an arbitrary document without errors and generates appropriate output. (50%)
- `answer` runs on a arbitrary document and a set of questions without errors and generates appropriate output. (50%)

Points for the final working system will be equally allocated between tasks. For each task, systems will be awarded points based on the quartile in which their scores fall:

Quartile	Points
Q1	0
Q2	1
Q3	2
Q4	3

A Making a Python File Executable on Linux

There is a frequent question with regard to the project: how do I make a Python file executable (similar answers will work for other scripting languages). We will address the specific case of Linux (and macOS) since the projects will be run on Linux servers.

Step 1: add an appropriate shebang line to beginning of the file. Since the default Python on the servers will be Python 2.7, to run a file with the Python 2.7 interpreter, the first line of your file should be as follows:

```
#!/usr/bin/env python
```

For Python 3, you should instead use:

```
#!/usr/bin/env python3
```

Step 2: change the permissions of the file so that it is executable by all users. At the command line, issue the following command (`$` is the command prompt):

```
$ chmod a+x thescript
```

Where `thescript` is the path to the file whose permissions you want to change.

That is it. You should now be able to run the script by typing the following command at the prompt:

```
$ ./thescript
```