CMPS 143 - Assignment 6

Due Monday, May 22, 11:55 PM

The next three assignments will be to design and build a question answering system. We will use stories from Aesop's Fables and the Blogs Corpus. For each homework you will receive new datasets with more difficult stories and questions.

We created questions and answer key for each story. Each question has a unique *QuestionID*, which is the *StoryID* followed by a dash and question number. For example, "blogs-01-1" means that this is question #1 pertaining to story "blogs-01". In some cases, the answer key allows for several acceptable answers (e.g., "Toronto, Ontario" vs. "Toronto"), paraphrases (e.g., "Human Immunodeficiency Virus" vs. "HIV"), varying amounts of information (e.g., "he died" vs. "he died in his sleep of natural causes"), or occasionally different interpretations of the question (e.g., "Where did the boys learn how to survive a storm?" "camping tips from a friend" vs. "their backyard"). When more than one answer is acceptable, the acceptable answers are separated by a vertical bar (|). Below is a sample answer key:

```
QuestionID: blogs-01-2
Question: What is the summit meeting named?
Answer: G20 summit
Difficulty: Easy
Type: Sch

QuestionID: blogs-01-3
Question: Where did the protest happen?
Answer: on a street | along the street where I work | right in front
of my store
Difficulty: Easy
Type: Story | Sch
...
```

Figure 1: The answer key for story blogs-01

The Task: Your must build a question answering (Q/A) system that can process a story and a list of questions, and produce an answer for each question. Your system must conform to the following input and output specifications, but other than that you can design your system however you want!

1 Input

Your Q/A system requires no command line arguments, but should be placed within the dataset directory. The dataset contains the following:

- story files named StoryID.story (e.g., "blogs-01.story")
- question files named StoryID.questions (e.g., "blogs-01.questions")
- answer files named StoryID.answer (e.g., "blogs-01.answer")
- Scheherazade realization files named StoryID.sch (e.g., "blogs-01.sch")

- constituency parses of the story named StoryID.par (e.g., "blogs-01.par")
- dependency parses of the story named StoryID.dep (e.g., "blogs-01.dep")
- constituency parses of the questions named StoryID.questions.par (e.g., "blogs-01.questions.par")
- dependency parses of the question files named StoryID.questions.dep (e.g., "blogs-01.questions.dep")

Your Q/A system should produce an answer for each question in each question file within the dataset directory, based on the corresponding story file.

Each question file will contain 4 lines for each question indicating the QuestionID, the question itself, a difficulty rating, and the file you should use to find the answer. The *Difficulty* ratings are based upon the methods required for extracting the answer. For this homework, all of the questions will have a difficulty rating of "easy". The *Type* field indicates which file you should use to answer this question, where "Story" indicates the .story file and "Sch" indicates the .sch file. You can and should use this information in your program. The question file will be formatted like Figure 1, except that there will not be an answer line. For example, it would look like this:

```
QuestionID: blogs-01-2
Question: What is the summit meeting named?
Difficulty: Easy
Type: Sch

QuestionID: blogs-01-3
Question: Where did the protest happen?
Difficulty: Easy
Type: Story | Sch
...
```

Figure 2: The question file for story blogs-01

2 Output

Your Q/A system should produce one single *Response File*, which contains the answers that your system finds for all of questions in the dataset directory. The output of your system should be formatted as follows, for example:

```
QuestionID: fables-01-2
Answer: a piece of cheese

QuestionID: fables-01-3
Answer:

QuestionID: fables-01-4
Answer: snatched
...
```

Figure 3: An example response file for story fables-01

IMPORTANT: There should be a *QuestionID* and *Answer* for every question in every story specified by the input file, in exactly the same order. Also, be sure to print each answer on a single line. If your Q/A system can't find an answer to a question (or your system decides not to answer a question), then just leave the answer blank (as done for *fables-01-3* in Figure 3).

To obtain a Satisfactory grade, your Q/A system can still be simple, but it should produce correctly formatted output and make a non-trivial, good faith attempt to answer some questions.

WARNING: You will be given the answer keys for some datasets, but your Q/A system is not allowed to use them to answer questions! For example, you can not just look up the answer to each question, or use the answer keys as training data for a machine learning algorithm. Your system must answer each question using general methods and you must use exactly the same system on both test sets. The answer keys are being distributed only to show you what the correct answers are, and to allow you to score your system's performance yourself.

3 Evaluation

The performance of each Q/A system will be evaluated using the F-measure statistic, which combines recall and precision in a single evaluation metric. Since Q/A systems often produce answers that are partially but not fully correct, we will score each answer by computing the *Word Overlap* between the system's answer and the strings in the answer key. Given an answer string from the answer key, your system's response will be scored for recall, precision, and F-measure.

As an example, suppose your system produces the answer "Elvis is great" and the correct answer string was "Elvis Presley". Your system's answer would get a recall score of 1/2 (because it found "Elvis" but not "Presley"), a precision score of 1/3 (because 1 of the 3 words that it generated is correct), and an F-measure score of .40.

Two important things to make a note of:

- This scoring measure is not perfect! You will sometimes receive partial credit for answers that look nothing like the true answer (e.g., they both contain "of" but all other words are different). And you may sometimes get a low score for an answer that seems just fine (e.g., it contains additional words that are related to the true answer, but these extra words aren't in the answer key). Word order is also not taken into account, so if your system produces all the correct answer words, but in the wrong order, it doesn't matter – your system will get full credit! Automatically grading answers is a tricky business. This metric, while not perfect, is meant to try to give your system as much partial credit as possible.
- The answer key often contains multiple answer strings that are acceptable for a question. Your system will be given a score based on the answer string that most closely matches your system's answer.

We have included this scoring function. To run the program:

```
perl score-answers.pl <response_file> <answerkey_file>
```

To run the evaluation on a single story:

```
perl score-answers.pl fables-01_my_answers.txt fables-01.answers
```

To run the evaluation on an entire dataset:

```
perl score-answers.pl train_my_answers.txt train.answers
```

4 Provided Files

We have provided you with several "stub" Python files to get you started. These cover how to read in the various files, how to process the data in those files, and how to use a parser and chunker. All of these files will be reviewed during class (and office hours, if desired).

The "easy" questions for this homework should be able to be answered with a simple string overlap or the use of regular expression patterns ('Where is A' → 'A is { in | under | at ...}'). It is not required to use the parser or chunker for this homework, however, these tools will be required for assignment 7.

5 What To Turn In

1. Your question-answering system .py file
2. Your response file that contains the answers to all the questions for all the stories. This should be called `train_my_answers.txt`