CMPS 143 - Assignment 6

Due Tuesday, Nov 12, 11:59 PM

The next three assignments will be to design and build a question answering system. We will use stories from Aesop's Fables, ROC stories corpus and MC test. For each homework you will receive more information about the text (of the story and questions) which can be used in the system and be required to return more exact answers.

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. For the first assignment we will only ask you to return the id of the sentence that the answer can be found in. 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

Download from Canvas and unzip the file. Your Q/A system requires no command line arguments, but should be placed in the same directory as the dataset. The dataset contains the following:

- story files named `hw6-stories.tsv` contains following columns,
  - *storyid*: story ID
  - *storytitle*: the title of the story (MC test doesn't have titles)
  - *sentenceid*: sentence ID. This is formatted as <storyid>_<sentence_index>, for example, "fables-01_2" means that this is the 3rd sentence (indexes start at 0) in the story "fables-01".
  - *sentence*: raw text of the sentence from the story
- question files named `hw6-questions.tsv`
  - *storyid*: the story ID of the story the question is from
  - *questionid*: unique question ID, which is the story ID *sid* followed by the question number. For example, "fables-01_Q1" means that this is question #1 pertaining to story "fables-01". "mc500.train.0_Q2" means that this is question #2 pertaining to story "mc500.train.0".
  - *question*: raw text of the question
- answer files named `hw6-answers.tsv`
  - *storyid*: the story ID
  - *questionid*: the question ID
  - *answer_sentenceid*: the sentence id of the sentence the answer should come from
  - *answer*: the correct answer to the question

The stories in `hw6-stories.tsv` are divided into sentences. You can choose to read the stories as single sentences or combine the sentences to get a complete text and read that into your system (though it needs to be able to read in the file we gave you unmodified). If you recreate the complete story you can use nltk.sent_tokenize(text) to divide the text into the same sentences we created.

The Answer Key. 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 in `hw6-answers.tsv`:

sid	qid	sentence_id	answer
fables-03	fables-03-20	fables-03_3	some poison | some of his poison
...	...	...	...

Project Files. Below is a listing of the files for this project as well as the file structure that is expected to run the starter code we provide for you:

```
+ data/
|   - hw6-answers.tsv
|   - hw6-questions.tsv
|   - hw6-stories.tsv
+ qa_engine/
|   - __init__.py
|   - base.py
|   - score_sentences.py
+ qa.py
+ baseline-stub.py
```

The command to run the Q/A system is `python3 qa.py`

2 Output

Your Q/A system should produce one single response file named `hw6-responses.tsv`, which contains the answer's sentenceid that your system finds for all of questions in `data/hw6-questions.tsv` with corresponding *questionid*. The output of your system should be formatted as in Table 1.

storyid	questionid	answer_sentenceid	answer
fables-01	fables-01-1	YOUR_ANSWERID	
fables-01	fables-01-2	-	
...	...	...	

Table 1: Sample of `hw6-responses.tsv` your system generates

IMPORTANT: Use "storyid" , "questionid" , "answer_sentenceid" and "answer" as your column names. There should be a row for every question. Also, be sure to print each answer on a single line. We use the actual questionid to index and score your answers, be sure to get it correctly. This is analogous to leaving the answer blank as shown in Figure 1. The column "answer" can be completely empty for this assignment. Please keep this column in the table so you can easily add your answer string for assignment 7 and 8.

Some of the questions have no answer in the text. These are marked with hyphens "-" in the file "hw6-answers.tsv". The correct answer for these questions is therefore a hyphen and your system should be able to output a hyphen if there is no answer to the question in the text as shown in Table 1. If for any other reason your system cannot find an answer either mark it with a hyphen or leave it blank. Either output will be processed properly by our evaluation script.
Also, each question should return a single sentence id. This should be the sentence with either all the relevant information (in most cases) or, in some cases, it might have the answer without the question. The sentenceid should always refer to the sentence which contains the text indicating the answer.
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: The answer keys given in *data/hw6-answers.tsv* are not allowed to use to answer questions! For example, you can not just look up the answer to each question. If you decide to follow a classification approach, then you can use the answers in the training examples to derive labels, as described in class. Your system must answer each question using general methods. 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 this assignment will be measured as an accuracy based on getting the correct sentenceid for the questions. Since even adjacent sentences will, in most cases, have no relevant information for answering the questions you need to get the exact correct sentence in order to have your answer marked as correct.
Running the scoring script. We have included this scoring function in `qa_engine/score_sentences.py`, and to run the scoring you can use command,

```
python3 qa_engine/score_sentences.py -answer data/hw6-answers.tsv -response
                            hw6-responses.tsv
```

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 and how to process the data in those files. The python files do the following:

`base.py`: Helper file, has no main function so does not run independently. Has a class, QABase, which reads in the stories and the questions and has functions to run the code and save answers. Do not strictly need to modify anything in this file but can make changes as desired.

`qa.py`: Main file. This runs the code in base.py. Need to modify get_answer(question, story) here to generate and output the sent_id and answer (however you could leave answer as empty string for now). The output of running this file is the results file described in Section 2.

`baseline-stub.py`: Potentially helpful starting point. An example of a baseline model which finds and prints the best sentence given a question_id as input. The question id is hard coded in the file, need to change it in the main function to experiment with other questions. You can chose to use elements from this stub or not, it is a potential starting point to work from.

`score_sentence.py`: Output evaluation. Prints the accuracy of your results file to the command line.

The questions in this assignment are of various levels of difficulty. Some of them are taken directly from the text while others use paraphrases and/or synonyms. These different questions might need to be handled differently in order to get the best accuracy in the output, so having multiple ways to handle different questions might be useful.

5 What To Turn In

Team submission. One submission per team, zip all files and it should include,

1. Your question answering system, name it `qa.py`
2. Your response file that contains the answer_sentenceids to all the questions for all the stories. This should be called `hw6-responses.tsv`
3. README including all your team members
4. Include all files required to run your program

Individual submission. Each individual should submit,

1. A text file includes your contribution