CSE 143 - Assignment 7

Due Thursday, November 21st, 11:59 PM

1 Introduction

This homework assignment will expand upon the question answering task introduced in the previous assignment. In Assignment 6 you were asked to find the correct sentence that contains the answer. And in this assignment you will focus on extracting the answer from that sentence. In order to accomplish the task we provide you a new dataset with additional information specifically, named entity recognition (NER), coreference resolution results from Stanford CoreNLP[1]. We also include a new scoring script to score your answers.

2 The Task

You must improve your 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 provided input and output specifications. Also, you must **improve your system by making use of NER and coreference information**. As for NER you could choose tools between Stanford NER that is included in the dataset, tools that are described in class. Other than these requirements you can implement your system however you want.
Reminder: While we are providing you the freedom to improve your own system., this does not mean that you can chose to make only minimum changes to your system from last week. That is, you cannot submit the same system from the last assignment again as your solution to this assignment.

3 Dataset

We have updated the files from Assignment 6 so you MUST download the new dataset for Assignment 7. Below is the description of the additional file and columns added to the new dataset,

- story file named **hw7-stories.tsv** contains **additional** columns,
  - *ner*: CoreNLP version of NER
- story file with coreference information named **hw7-stories-coref.tsv** contains columns,
  - *storyid*: story ID
  - *storytitle*: the title of the story (MC test doesn't have titles)
  - *coref*: CoreNLP version of coreference resolution
- questions file named **hw7-questions.tsv** contains **additional** columns,
  - *ner*: CoreNLP version of NER
- We also refined some of the questions and answers so please do re-download the new dataset.

[1] `stanfordnlp.github.io/CoreNLP`

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 `hw7-answers.tsv`:

storyid	questionid	answer_sentenceid	answer
fables-03	fables-03-20	fables-03_3	some poison | some of his poison
...	...	...	...

Reminder: Note that you are NOT allowed to use the *answer_sentenceid* in `hw7-answers.tsv` as a method to get the correct sentence.

4 Provided Files

The new dataset is located in the `data` directory. We have also made changes to the `qa_engine` framework so that it is compatible with the new dataset.

- We modified the input filenames to point to new dataset in `qa_engine/base.py`
- We included functions to read in the new dataset described in Section 3.
- We added a new scoring function to score your answers `qa_engine/score_answers.py`.
- We included `baseline-stub-word2vec.py` as a demo of using word embedding to rank sentences of the story.

File structure. 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/
|   - hw7-answers.tsv
|   - hw7-questions.tsv
|   - hw7-stories.tsv
|   - hw7-stories-coref.tsv
|   - glove-w2v.txt
+ qa_engine/
|   - __init__.py
|   - base.py
|   - score_sentences.py
|   - score_answers.py
+ qa.py
+ baseline-stub-word2vec.py
+ word2vec_extractor.py
```

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

5 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.
- Command to run the scoring function,

```
python3 qa_engine/score_answers.py -answer data/hw7-answers.tsv
-response hw7-responses.tsv
```

6 What To Turn In

Team submission: Pick one team member to make the team submission. The team submission should be a zip file with the last names of each group member. For example, if the group members were Jane Smith and John Doe then their team submission file would be named `Smith_Doe_6.zip`. The team submission zip file needs to contain the following files:

1. Your question answering system, name it `qa.py`.
2. Your response file that contains the answers to all the questions for all the stories. This should be called `hw7-responses.csv`
3. README including all your team members names and any notes, if necessary, to the grader on how to run your program.
4. Include all files required to run your program other than `qa_engine`.

Individual submission: Each individual should submit the following,

1. Write down who is on your team, and who is responsible for turning in the code.
2. Write a summary of what got done during this phase of the project. This should be represented by a list of tasks and how they were completed.
3. Write down your individual contribution during this phase of the project.
4. Write down the contribution of your other team members for this phase of the project.