Project

LIN/CSE 467/567

Due by 11.59pm on May 6th[*]

1 Grading

This project is worth 21 points (distributed across 7 tasks), and can be solved in groups of 1 or 2 students. All students should upload their project, clearly identifying all of the authors. Students can also disclose their personal contributions to the projects.

To submit the project, just upload all the code (single file or multiple files, uncompressed) to UBLearns. We should be able to run your code and obtain, for full credit to be awarded. See the syllabus for information about plagiarism concerns.

2 Description

The goal of this project is to construct a small-scale question-answering system that can give exact answers to queries of arbitrarily complex sentences.

The system can be implemented in pure Python or via with COLAB (you can also use bash commands, spaCy, StanfordNLP, Stanza, NLTK, Pytorch, etc.). It must be able to cope with three main kinds of input:

- **Declarative sentences**, for example:

  (1) a. A tall skinny man coughed.
  b. Sam bought a black sedan in the dealership around the corner.
  c. The employee who assisted Mia left the dealership.
  d. Robin was interrupted by a drunk sailor with a mustache.

  [3 points]

- **Yes-No Questions**, for example:[1]

  (2) a. Did a tall skinny man sneeze?
  b. Did Sam buy a black sedan in the dealership around the corner?
  c. Did the customer who called the dealership write a check?
  d. Was Robin interrupted by a drunk sailor with a mustache?

[*]Submissions will be accepted without penalty until May 9th 11.59pm (hard deadline).

[1]Note that declarative sentences include actives and passives.

[3 points]

- **Wh-Questions**, for example:

(3) a. Who sneezed?
b. Who called the owner of the car?
c. What did the owner of the dealership see near the door?
d. What was Robin interrupted by?

[3 points]

You must implement a CFG with First-Order Logic and $\lambda-$terms that has **at least** 20 common nouns, 10 proper names, 10 adjectives, 10 intransitive verbs, 10 transitive verbs, 10 prepositions (passive 'by', and also non-vacuous prepositions such as 'near', 'under', 'above', 'by', etc. which denote spacial relations), the complementizer 'that', 2 auxiliary verbs, and 2 *wh*-pronouns ('who' and 'what').

Your system must be able to parse sentences that have the same structure as those in the example sentences above, and yield their respective semantic representations. The system should be able to parse a potentially infinite number of sentences because of the many recursive rules drawn from the mini-grammar discussed in the slides. You may, if you wish, augment the syntactic rules with new ones, but they must be able to yield well-formed First-Order Logic representations.

[3 points]

2.1 Information flow

The general sequence of steps of your QA system must be as follows.

1. An input sentence (a string, not a list of tokens) is specified (either via a user prompt, COLAB dialogue box, or simply a Python-internal variable declaration). The input must have a punctuation mark (a period or a question mark).

   E.g. *A man entered the dealership.*

2. A tokenizer and a lemmatizer are ran over the normalized input, to produce a list of uninflected tokens.

   E.g. [*a, man, enter, the, dealership*]

3. The input is parsed by a NLTK CFG, augmented with features and $\lambda-$terms, as discussed in the semantic section of the course.

   Most verbs in the grammar can be listed without inflection (e.g. no lexical entry for *entered* or *enters*, just one lexical entry for *enter*), with the goal of keeping the grammar as small as possible.

   Since there is only one lexical entry per verb, this means that the lexical entry for any TV (e.g. 'see') should interact with rules of the grammar to allow for with active declarative sentences (e.g. "Tom saw a blue car"), passive declaratives (e.g. "A blue car was seen by Tom"), active wh-questions (e.g. "What did Tom see?"), passive wh- questions ("What was seen by Tom?"), active Y-N questions (e.g. "Did Tom see a blue car?"), passive Y-N questions (e.g. "Was a blue car seen by Tom?"), active subject relatives (e.g. "Robin saw the customer who bought the Ferrari"), passive subject relatives (e.g. "Robin bought a car that was seen by Tom"), etc. All of these cases involve exactly the same lexical entry for the verb, combined with various general rules that license actives, passives, interrogatives, subject relatives, etc., as discussed in the slides and lectures. This keeps the grammar very small, since all the complexity comes from how the rules interact with lexical entries.

4. The first First-Order Logic form output by the grammar is fed into a model checker according to the type of input sentence as detailed below.
   (a) If the sentence is declarative, then the model checker just needs to react to whether it agrees with the assertion or not. For example:[2]

   INPUT: *A man leave the dealership.*
   OUTPUT: `I know`.
   INPUT: *A cat left the dealership.*
   OUTPUT: `I don't think so`.

   [3 points]

   (b) If the sentence is a Yes-No Question, then the model checker just needs to determine whether the query is true or not. For example:

   INPUT: *Did a tall man leave the dealership?*
   OUTPUT: `Yes`.
   INPUT: *Did a brown cat leave the dealership?*
   OUTPUT: `No`.

   [3 points]

   (c) If the sentence is a Wh-Question, the model checker needs to determine what the name of the entity that is the answer to the query is, if it has a name (and if there is an answer). For example:[3]

   INPUT: *Who left the restaurant?*
   OUTPUT: `Tim`.
   INPUT: *What did Mary see?*
   OUTPUT: `Felix`.
   INPUT: *Who did the customer who insulted Robin complain to?*
   OUTPUT: `Sorry, I don't know their name`.
   INPUT: *What was Alex interrupted by?*
   OUTPUT: `Nothing`.

   [3 points]

It is up to you to create a model that makes some sentences true and some sentences false, as illustrated above, for each of the three types of input sentence. You are free to create your model as you wish.

[2] The lexical representation of the definite determiner *the* should be analogous to other existential determiners. The uniqueness constraint associated with *the* is arguably pragmatic (presuppositional) in nature, and therefore we need not concern ourselves with it here.

[3] For simplification, assume that *what* and *who* impose no semantic constraints of their own, so that *Felix, our cat/neighbor* is a licit answer to *What/Who made that noise?*