UNIVERSITY OF TÜBINGEN, DEPARTMENT OF GENERAL LINGUISTICS

Homework 3: Syntactic Complexity

Xiaobin Chen, Zarah Weiss, and Detmar Meurers

Due December 22th, 2019, 23:59

1 Extracting relative clauses from a learner corpus

In this assignment, you will implement a syntactic complexity analyzer that extracts Relative Clauses (RC) from a list of learner produced sentences. You will need to use parsers (either constituency or dependency parsers) for the task. The sentences in the list have also been manually annotated on whether they contain RCs. The manual annation will not be given to you before the submission of the assignment, but instead will only be used to evaluate the performance of your RC extractor. We will calculate the precision, recall, and F score of your tool against the human annotated gold standard.

1.1 Input and Output Requirements

Your program should take the following input arguments (in this order):

1. An input file containing all the sentences to be analyzed. Each line of the text file contains one sentence.
2. The name of the output csv file to which the results should be saved.

Your program should output a single csv file (as specified by the user in the input) with the following columns:

sent_id the id of the sentence as from the input file.

sentence the sentence text as from the input file.

has_rc whether the sentence contains relative clauses. If it contains RCs, put a value of 1, otherwise 0.

RCs a list of the relative clauses contained in the sentence in the format *"clause 1 text", "clause 2 text", ...* separate the clauses with commas and quote each clause with double quotes. Please remember to escape the double quotes in the clause texts.

Please also use TABs as column separators.
Not required but would be nice to have: If you would like to challenge yourself, also try to extract the following information from the identified RCs: head noun, the relativizer (zero if it

is a zero relative), the grammatical role of the head noun in both the main and relative clauses (subject, direct object, indirect object, etc.). If you implement these functions, list each RC in their own lines but keep track of the sentence id, because there might be multiple RCs in one sentence.

1.2 Resources

You may write your code either in Java or in Python 3. Use one of the following APIs for sentence segmentation and tokenization:

- **OpenNLP** version 1.9.1 using the corresponding off-the-shelf model for English[1]
- **Stanford CoreNLP** version 3.9.2 using the corresponding off-the-shelf model for English[2]
  - Note: **don't** use the model English (KBP)
- **NLTK** version 3.4.5[3]
  - Use the nltk.tokenize.punkt sentence segmentizer and
  - the nltk.tokenize.word_tokenize tokenizer with the pretrained English model.
- **spaCy** version 2.2[4]
  - use the English model en_core_web_sm

1.3 Submission

Submit your executable code via moodle in the format specified under *General Submission Requirements* at the end of this document. The submission deadline is

- **December 22th, 2019 23:59**

1.4 Hints

- It is possible to realize the task with either constituency or dependency parsers. With constituency parsers, you will need to write the Tregex rules for extracting relative clauses. With dependency parsers, read about dependency relations to find the relation that identifies RCs and find ways to extract the span of a relation from the semantic graph parses.
- Before you write any Tregex or dependency rules, it is always a good idea to inpect the parse results with some graphical tools like CoreNLP's Web demo or Grammarscope.
- For a task like this, the tool can not be 100% accurate, but it is important to know what you can extract and what you might miss so that when interpreting the results from your analysis tool, you also know the limitations.

2 Use your Code on the given Data Set

Run your code on the following test data and include the resulting csv under the name **results-task3.csv** before you submit.

- The sentences to be analyzed are included in the file **list_of_learner_produced_sentences.csv**.

The data is a set of sentences extracted from EFCAMDAT (`https://corpus.mml.cam.ac.uk/efcamdat2/public_html/faq/`), a large scale corpus of learner produced texts from an online English learning system.

[1] `https://opennlp.apache.org/download.html`
[2] `https://stanfordnlp.github.io/CoreNLP/download.html`
[3] `https://www.nltk.org/install.html`
[4] `https://spacy.io/`

3 Know Your Code

You need to be able to explain your pipeline in a few words in front of the class. We will choose a student at random to elaborate on their code. Relevant information are for example:

- Which components did you use in your pipeline?
- In which order are they executed?

4 General Submission Requirements

4.1 Submission

Homework can only be submitted via moodle before the end of the respective deadline. Your submission must always include a README file named **README.txt** including the following information:

- Full names and email addresses of all students who contributed to and want credit for this homework.
- The programming language use used
- A full list of the NLP APIs you used
- The name of your main file that should be called for execution of your code.
- A preliminary summary of the performance of your tool. Manually inspect some sentences to get a feel of how well your tool is doing in extracting RCs from learner sentences. Summarize what your tool does well and which types of RCs tend to be missed or misjudged as RCs by your tool. One or two paragraphs should suffice.

You may also submit other files (e.g. plots, animations, etc.) that you think will help us understand your points.

4.2 Programming Languages

- We assume that you program in either Java or Python 3. If you have a good reason to use another programming language, talk to us about it early as possible.
- If you use Java, make sure to use Maven to handle your dependencies (`https://maven.apache.org/`). Submit an executable jar file as well as your source code.
- If you use Python 3, make sure to use a virtual environment to handle your dependencies. If you are not familiar with virtual environments, you may finde the following links useful:
  - `https://www.geeksforgeeks.org/python-virtual-environment/`
  - `https://docs.python-guide.org/dev/virtualenvs/`
- Make sure that your code can be executed. If we cannot run your code, it counts as a failure.