LING 807- Assignment 1

Name (alphabetical)	Student ID	Contribution #1	Contribution #2	Other Contributions
Rynearson, Amber	301616904	B write-up	A notebook	
Hashemi, Romina	301630807	B+ write-up	Wrote the function for indirect quotes with reporting verbs, and designed the examples/tests.	Debugged the function for indirect quotes with no reporting verbs.
Tammy Chen	301472839	Test of Approach#2	Indirect Quotes without reporting verbs notebook	

B:

Approach #1 (Regular Expressions)

The first approach (regular expressions) essentially searches for something in the text (in this case, the quotation marks) and prints out anything in between the quotation marks. It works similarly to the find-and-replace feature in many different software, including Word and Google Docs. The following table describes the output, the number of actual direct quotes, as well as the number of identified quotes that were not actually quotes.

File Name	Output	Actual DQs	Captured non-DQs
File_1	3	9	0
File_2	2	5	2
File_3	10	19	9
File_4	0	9	0
File_5	3	6	0

The regular expression approach returned far fewer quotes than actually present in the files; for example, the first file (File_1.txt) returned 3 direct quotes (DQs) out of a total 8 DQs within text. File_2 and File_3 both returned words within quotes that were not direct quotes, but instead “false-positive,” or words encased in quotation marks that are not being spoken by someone. For example:

(1) Ahead of the nomination vote Saturday, Eduljee told a crowd of about 100 party members gathered at a banquet hall in Burnaby that the "whole country" will be watching the vote in Burnaby South's byelection.
(2) Federal officials in the U.S. have even issued a "black box warning" — the most severe of all warnings — about the risks of combining certain opioids with benzodiazepines.

This approach likely failed to identify all direct quotes because it was simply searching for the quotation marks, and failed to account for the other parts that make a quote direct. In addition to quotation marks, a direct quote includes a verb (often directly before or after the quote), a speaker of the quote, as well as potential punctuation with the quotation marks. There may have also been too much space in between the quotation marks for the sentence to register as a quote. To successfully identify all quotes (or at least more quotes), these aspects should be incorporated into this approach.

Approach #2 (SpaCy's Matcher)

SpaCy's Matcher allows the user to apply a rule-based pattern to identify specific things (ex. tokens) in a text. This particular approach starts by identifying the proper nouns; although this step is optional, it may provide guidance in identifying the speakers of the quotes. The second, required part using the Matcher, creates a pattern to identify the text content between quotations. The pattern is essentially the Matcher seeking a quotation mark, at least one word of text, a potential punction mark (such as a comma), and a final quotation mark. The following table describes the output, the number of actual direct quotes, as well as the number of identified quotes that were not actually quotes.

File Name	Output	Actual DQs	Captured non-DQs
File_1	3	9	0
File_2	3	5	2
File_3	12	19	9
File_4	2	9	0
File_5	3	6	0

Again, this approach falls short of identifying all DQs in the files. However, there is a slight improvement as File_2 outputs one DQ (in addition to the non-DQ outputs) and File_4 outputs 2 DQs where the RegEx initially returned nothing. However, this approach fails to identify every direct quote within a given text and captures things that are not actual quotes (such as the earlier examples of "whole country" and "black-box warning" in (1) and (2) respectively).

This approach fails in similar ways to the RegEx approach, as the fact that direct quotes are made up of more than just text between quotation marks is not considered in these codes. To fix this method, similar considerations as listed above for RegEx, should be taken into consideration, such as including the presence of an entity/speaker and the presence of a verb either directly before or after the quotation marks.

Approach #3 (Implemented Version)

This approach uses QuotationTool to extract direct quotes from the files, and its code is modified from the Gender Gap Tracker. The accuracy of this tool is higher than that of the previous two approaches, and the accuracy ranges from 76.1% to 83.7% depending on the difficulty of the text (Asr et al., 2021). The output consists of a .csv file that is filled with the quotes, as well as other information including the speaker, number of quotes, verbs, and entities, among other information. The following table illustrates what was captured by the tool:

File Name	Output	Actual DQs	Captured non-DQs
File_1	15	9	5 (indirect quotes)
File_2	14	5	9 (indirect quotes)
File_3	32	19	13 (indirect quotes)
File_4	21	9	12 (indirect quotes)
File_5	14	6	8 (indirect quotes)

Unlike the previous approaches, this tool does capture all of the direct quotes. However, it also captures indirect quotes and misconstrues some of the text as a quote when it is not. It also did not always capture the entirety of a quote, see the following, where (1) is the captured quote and (2) is the entire quote:

(3) This order takes effect immediately and will remain in effect as long as necessary [file 4]
(4) "My department has issued a Ministerial Order under the Railway Safety Act to all railway companies mandating the use of handbrakes should a train be stopped on a mountain grade after an emergency use of the air brakes. This order takes effect immediately and will remain in effect as long as necessary"

This approach also split up some of the direct quotes, returning 2 outputs instead of what should have been 1. See (5) and (6)

(5) We're not giving up, but
(6) it feels really overwhelming to think about what this means and what they're trying to do to us right now

(5) and (6) are one quote within quotation marks; however, it's really just one quote and it should have been identified as such.

Comparing the approaches

All three approaches fell short of extracting direct quotes with 100% accuracy in one way or another, although the implemented version based on the Gender Gap Tracker is the most accurate overall. This approach pulled out all of the direct quotes as well as indirect quotes, speakers, and more, even if there were some issues and/or inaccuracies in output. Conversely, both the RegEx method and SpaCy's Matcher failed in similar ways, failing to capture many of

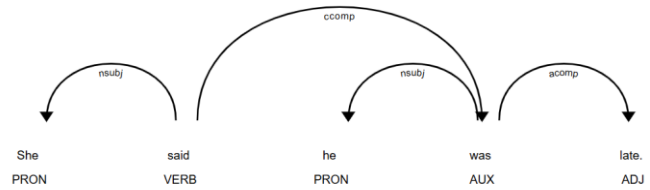
the direct quotes and instead often catching phrases that were not actually quotes (see (1) and (2) above). Both of these methods failed to account for aspects of direct quotes (such as the related verbs and speakers) that were incorporated into the implemented method. Overall, these methods demonstrate the difficulty in accurately identifying direct quotes in data.

B+:

Indirect quotes can take many syntactic forms. For starters,

(1) She said he was late.	(2) She told us she was late.
(3) She said that she was late.	(4) She told us that she was late.
(5) She requested to leave.	(6) She asked us to wait for her.
(7) That she was late is all he told us.	(8) What she told us was that he was late.
(9) According to her, he was late.	

The spaCy dependency tree for (1) is as follows:



The pronoun *she* is identified as a nominal subject (NSUBJ), and the verb *was* is identified as a clausal complement (CCOMP) with subtree, “he was late.” These classifications correspond to speaker and the indirect quote of the sentence. For (2), NSUBJ is *she* and the subtree of the clausal complement is “she was late.” For (3) and (4), NSUBJ is *she* and the subtree of the clausal complement is “that she was late.” In sentences (5) and (6), the speaker *she* is identified as NSUBJ and the verbs *wait* and *leave* are identified as open clausal components (xcomp): the subtrees for the open clausal complements are “to leave” and “to wait for her,” respectively. In (8), the speaker *she* is an NSUBJ within the “what she told us” phrase, and the first copula *was* is the ROOT, so the second copula *was* is a CCOMP, with subtree, “he was late.”

In order to implement the method, we first check if each sentence includes a quotation mark. If it does, it is disqualified due to being a direct quotation. Then, we compile a preliminary list of reporting verbs—*say*, *mention*, *claim*, *tell*, *argue*, *request*, *ask*. Next, we check whether a token is an NSUBJ, and whether its head is a reporting verb; if both conditions are met, the token is assigned the role of speaker. Now, if a token has a reporting verb or a copula as its head and has dependency CCOMP or XCOMP, then its subtree would be an indirect quote. Once we've identified both a speaker and a quote, we print them out.

This method will fall short because it does not account for the indirect quotation patterns of (7) and (9): one way to resolve this might be to use regular expressions or spaCy’s pattern matching to extract sentences that contain “according to.” It also needlessly excludes sentences that enclose a word in quotation mark while not being a direct quote, or sentences that combine an indirect and a direct quote. One fix might be to compare the quotes with those found by the direct quote finder, in order to eliminate overlap. This method also does not work when an indirect quote is split over multiple sentences or for embedded indirect quotes: it only identifies “that Bob said he was leaving” in the sentence, “Alice claims that Bob said he was leaving.”

A:

*See Notebook

I (Amber) wrote and added patterns to the matcher to pull out more quotes and the speakers. This was somewhat successful, as it pulled out a few more quotes than the original matcher pattern, as well as a few speakers for each of the files. This code also identified the entire/multiword entities in the text (which includes the speakers of the quotes), as well as the pronouns, both of which can be the speaker/subject of a quote. While this approach captured more quotes and speakers, it still did not capture all and occasionally captured false positives. The following table describes the results of this code.

File Name	Output	Actual DQs	Captured non-DQs	Captured speakers
File_1	4	9	0	2
File_2	3	5	1	0
File_3	10	19	4	2
File_4 *	5	9	0	2
File_5	4	6	1	2

*Most successful file

A+:

See notebook.

Works Cited

Asr FT, Mazraeh M, Lopes A, Gautam V, Gonzales J, Rao, P., & Taboada, M. (2021) The Gender Gap Tracker: Using Natural Language Processing to measure gender bias in media. PLOS ONE 16(1): e0245533. https://doi.org/10.1371/journal.pone.0245533