Manual Annotation Guidelines

Introduction

Package files & folders

Instructions for annotating each news article
- Overview
- 1) Annotating quotations for an article
- 2) Annotating people for an article
- 3) Annotating sources for an article
- 4) Annotating the type of news article
- 5) Comments

Examples and unusual cases
- Direct quote with speaker and verb
- Indirect quote / paraphrase
- Paraphrase with direct quote within it
- Quote with no verb
- Floating quote
- Verbs with auxiliaries
- Unusual quotative verbs
- Quote with an extra complement
- Quote with a possessive and noun

Questions and contact information

Appendix I: Introduction to JSON
- Overall structure
- Handling quotation marks
- Other quotation marks
- Checking your file

Introduction

This document describes:

A. The files included in this package, such as the main .xlsx data table, the folder structure, the texts of the news articles, and so on.
B. The annotation guidelines to extract the following information from each article
   a. All quotations by any entity (person, organization, group, etc.)
   b. Names and genders of all people mentioned by name in the article
   c. Names and genders of all people (b) who are also quoted (a)
   d. Genre of news / type of article

Package files & folders

1) "AnnotationTable_byArticle.xlsx"
This is the main source file, which provides all the information that we have automatically extracted for each article scraped from news websites. This table also includes a set of named columns with no content, where the annotators are supposed to fill in values.

2) "RawTexts" folder
This directory includes the raw text of each news article as a separate txt file (these texts are also provided in the data table within the body column). Annotators will be asked to use these when they perform annotation of certain fields: speaker, verb and quote in extracting quotes from the text of an article. Annotators should not make any changes to the content of the RawTexts folder. This should be treated as read-only.

3) "TokenizedTexts" folder
This directory includes the tokenized text of each news article as a separate txt file. If you open one of these files, you will see that each and every token (word) of the text is associated with a range of character indices. For example, *corn* might have the character range (388,392) because it spans 4 characters - from 388 to 392 (not inclusive of 392).
Annotators will be asked to use these indices when they perform annotation of certain fields: speaker_index, verb_index and quote_index when they extract quotes from the text of an article. Annotators should not make any changes to the content of the TokenizedTexts folder. This should be treated as read-only.

4) "OutputJsonFiles" folder
An empty folder, which will contain the output JSON files created by the annotators (following the template of "GoldSampleExtractedQuotes.json").

5) "GoldSampleHighlightedText.pdf"
This file includes a highlighted version of the text of the first news article listed in AnnotationTable_byArticle.xlsx. We have annotated this one text from the data table so the annotators can learn from our coding.

6) "GoldSampleTokenizedText.txt"
This is the tokenized text file of the sample article that we used for annotating speaker_index, verb_index and quote_index when we were annotating the sample text.

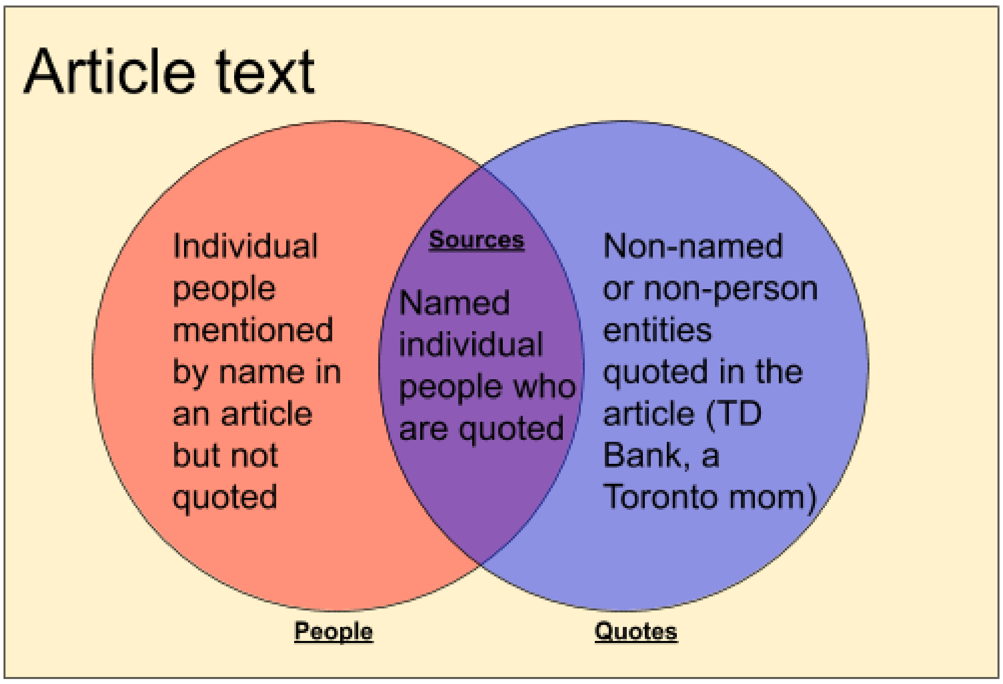
7) "GoldSampleExtractedQuotes.json"
This is the output of our annotation for the sample article, which includes all quotes, their speakers, and verbs. By looking at the GoldSampleTokenizedText.txt, we also figured the quote_index, speaker_index and verb_index of each quote and annotated them in this JSON file as well.

Important note: In order to open/read/write files with .txt or .json extensions, please use a simple text editor application such as Notepad++ on Windows systems, or Textwrangler or TextMate on Mac systems. For .xlsx files, please use a spreadsheet application such as Microsoft Excel.

Instructions for annotating each news article

Overview

The following Venn diagram represents the three components of an article that take the most effort to annotate. We are interested in all three of these categories.


Article text
Individual people mentioned by name in an article but not quoted
Sources
Named individual people who are quoted
Non-named or non-person entities quoted in the article (TD Bank, a Toronto mom)
People
Quotes

The annotation of the blue circle, i.e., the Quotes, is outlined in the first subsection. This involves capturing the names of the entities quoted, the words they say, and how they are quoted (e.g., “Person X **says** fact Y”, “Person A **claims** fact B”)

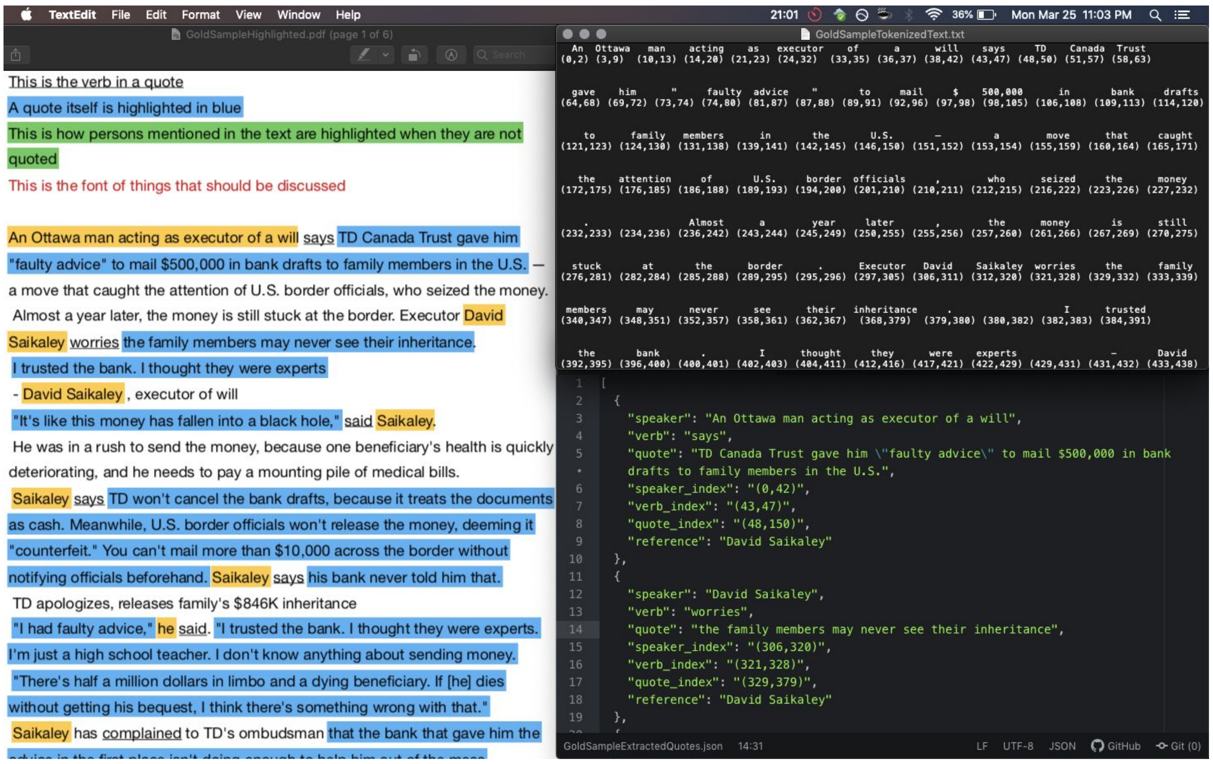
The annotation of the red circle, i.e., the People, is in the second subsection. We only need their names and genders, as they are not quoted.

The annotation of the purple intersection, i.e., the Sources, is explained in the third subsection. We have already captured all the quotes in the first subtask of annotation, so now all that is left is to list out the names of the People who were quoted, and try to establish their genders as well.

1) Annotating quotations for an article

(output: a JSON file (per article) in the "OutputJsonFiles" **folder)**

In order to understand the annotation task before you actually start the work, you should first open all GoldSample files in front of you on the same screen. Please see the screenshot of my monitor in this setup below.


TextEdit File Edit Format View Window Help
GoldSamplePDFHighlighted.pdf (page 1 of 6)
Search
21:01 36% Mon Mar 25 11:03 PM
GoldSampleTokenizedText.txt
GoldSampleExtractedQuotes.json 14:31 LF UTF-8 JSON GitHub Git (0)

This is the verb in a quote
A quote itself is highlighted in blue
This is how persons mentioned in the text are highlighted when they are not quoted
This is the font of things that should be discussed

An Ottawa man acting as executor of a will says TD Canada Trust gave him "faulty advice" to mail $500,000 in bank drafts to family members in the U.S. — a move that caught the attention of U.S. border officials, who seized the money.
Almost a year later, the money is still stuck at the border. Executor David Saikaley worries the family members may never see their inheritance.
I trusted the bank. I thought they were experts
- David Saikaley , executor of will
"It's like this money has fallen into a black hole," said Saikaley.
He was in a rush to send the money, because one beneficiary's health is quickly deteriorating, and he needs to pay a mounting pile of medical bills.
Saikaley says TD won't cancel the bank drafts, because it treats the documents as cash. Meanwhile, U.S. border officials won't release the money, deeming it "counterfeit." You can't mail more than $10,000 across the border without notifying officials beforehand. Saikaley says his bank never told him that.
TD apologizes, releases family's $846K inheritance
"I had faulty advice," he said. "I trusted the bank. I thought they were experts. I'm just a high school teacher. I don't know anything about sending money.
"There's half a million dollars in limbo and a dying beneficiary. If [he] dies without getting his bequest, I think there's something wrong with that."
Saikaley has complained to TD's ombudsman that the bank that gave him the

```
  An   Ottawa   man    acting    as   executor    of      a      will    says     TD    Canada  Trust
(0,2) (3,9)  (10,13) (14,20) (21,23) (24,32)  (33,35) (36,37) (38,42) (43,47) (48,50) (51,57) (58,63)

  gave    him      "     faulty  advice     "      to      mail     $    500,000     in      bank    drafts
(64,68) (69,72) (73,74) (74,80) (81,87) (87,88) (89,91) (92,96) (97,98) (98,105) (106,108) (109,113) (114,120)

   to      family   members     in       the      U.S.       —         a       move      that     caught
(121,123) (124,130) (131,138) (139,141) (142,145) (146,150) (151,152) (153,154) (155,159) (160,164) (165,171)

   the    attention    of       U.S.     border  officials     ,        who      seized     the      money
(172,175) (176,185) (186,188) (189,193) (194,200) (201,210) (210,211) (212,215) (216,222) (223,226) (227,232)

    .                 Almost      a        year     later       ,        the      money      is       still
(232,233) (234,236) (236,242) (243,244) (245,249) (250,255) (255,256) (257,260) (261,266) (267,269) (270,275)

  stuck      at        the     border       .     Executor   David   Saikaley  worries     the      family
(276,281) (282,284) (285,288) (289,295) (295,296) (297,305) (306,311) (312,320) (321,328) (329,332) (333,339)

 members     may      never      see      their   inheritance    .                   I      trusted
(340,347) (348,351) (352,357) (358,361) (362,367)  (368,379)  (379,380) (380,382) (382,383) (384,391)

   the      bank        .        I      thought     they      were    experts                 —       David
(392,395) (396,400) (400,401) (402,403) (404,411) (412,416) (417,421) (422,429) (429,431) (431,432) (433,438)
```

```json
[
  {
    "speaker": "An Ottawa man acting as executor of a will",
    "verb": "says",
    "quote": "TD Canada Trust gave him \"faulty advice\" to mail $500,000 in bank
    drafts to family members in the U.S.",
    "speaker_index": "(0,42)",
    "verb_index": "(43,47)",
    "quote_index": "(48,150)",
    "reference": "David Saikaley"
  },
  {
    "speaker": "David Saikaley",
    "verb": "worries",
    "quote": "the family members may never see their inheritance",
    "speaker_index": "(306,320)",
    "verb_index": "(321,328)",
    "quote_index": "(329,379)",
    "reference": "David Saikaley"
  },
```

Learning from GoldSample annotation of quotes.

In the pdf file you can see a color help. Look at the yellow highlights, which mark the speakers of the quotes in the text; the blue ones show the text span of the quoted speech, and the underlined words are the verbs used for quotations.
The JSON file with a black background in the screenshot includes the annotations for this sample text; you are supposed to create similar JSON files for other news articles in the dataset. Now have a look at the tokenized file, GoldSampleTokenizedText.txt, which is a guide for the character indices of each and every token (word) of the text.

As an annotator, you are supposed to read the text, find quotes and annotate all the requested information for each quote in a JSON file with the help of both the raw text (the content of the body column in the AnnotationTable_byArticle.xlsx file) and the tokenized version of the text provided in a separate file (the content of the TokenizedTexts folder). If you have not heard of or worked with the JSON file format before, please read the section on JSON in the appendix.

Each record in your output annotation file should have the following format, and records should be separated by commas.

```
{
    "speaker": "...",
    "verb": "...",
    "quote": "...",
    "speaker_index": "(...,...)",
    "verb_index": "(...,...)",
    "quote_index": "(...,...)",
    "reference": "..."
}
```

The content of the "speaker", "verb" and "quote" should be copied and pasted from the raw text of a news article, and the content of the "speaker_index", "verb_index" and "quote_index" should be annotated by getting help from the tokenized text. Finally, the content of the "reference" field needs to be the most complete name of the person or the speaker of the quote that you can infer from the context. The preferred form of a reference is a full name, including the first and last name of the person. For organizations such as banks etc. you could use any form, e.g., "TD bank" (see the example annotations in the sample file). Make sure you read all of the samples in the gold JSON file to learn about the annotation of different cases, such as cases where the speaker or verb of the quotations are empty.

- In case the speaker of a quote is empty but the person can be inferred implicitly from the text, you should write the name of that person in the "reference" field, but leave the "speaker" and "speaker_index" fields empty.
- For empty verbs, you should leave both the "verb" and "verb_index" empty.

Please read the sample text and find the corresponding lines for the following annotation examples:

```
{
    "speaker": "David Saikaley",
    "verb": "",
    "quote": "I trusted the bank. I thought they were experts",
    "speaker_index": "(433,447)",
    "verb_index": "",
    "quote_index": "(382,429)",
    "reference": "David Saikaley"
},
{
    "speaker": "",
    "verb": "",
    "quote": "\"I would have sought out a different method for sending the
money.\"",
    "speaker_index": "",
    "verb_index": "",
    "quote_index": "(4992,5059)",
    "reference": "David Saikaley"
}
```

For the *speaker_index*, *verb_index* and *quote_index* fields in each record, use the tokenized text to find the range of indices (*start*,*end*) of the entire speaker, verb or quote you are considering. For example, in the first quote of the above example, the speaker is David Saikaley. In the context of that quote, the speaker's name consists of two tokens - 'David' and 'Saikaley'. 'David' has indices (433,438) and 'Saikaley' has indices (439,447) so 'David Saikaley' has a range from the start of David to the end of Saikaley, i.e., (433, 447).

The boundary of each quotation should be taken maximally, including any quotation marks, if they are present. In such cases, ensure that the *quote_index* field indices line up with your *quote* field, i.e., your quote index must start at the starting index of the quotation mark, not the first word inside the quote. If the quote is a paraphrase preceded by the word "that," include the word "that" in the boundaries of the quotation.

Please note that **multiple quotations of the same person should be each annotated separately.** Do not join two consecutive quotations if the first quotation is finished with a quotation mark (") or a new line before the second quotation starts, or if any word external to the speech appears in between the two quotation pieces.

When you decide to start the annotation of quotations, open the "AnnotationTable_byArticle.xlsx" file and start with the second row of data (the first row already includes the gold annotation of the text we just discussed). You can open the text of the news article under investigation from the RawTexts folder in a text editor of your choice, then, on the side, open the corresponding tokenized version (use the _id column of the table to find the raw and tokenized text file names). Now you're ready to go: start reading the text and annotate the first quotation you find in a JSON file in a similar format to the sample JSON file! Make sure you read the right token index for the speaker, verb and the quote itself.

At the end of annotation, the JSON file needs to be saved with the same name as the name of the raw and tokenized text files, but with a '.json' ending, and in a separate folder called "OutputJsonFiles". After finishing each JSON file, please copy its content and paste it into the corresponding cell within the AnnotationTable_byArticle.xlsx file (in the quotations column).

Please note that if you are using a quotation mark character within a line of JSON that already has quotation marks, you must put a backslash in front of it. Refer to the first quote of the gold JSON annotation for an example of this. If you want to check that your JSON is valid, you can use a JSON validator. With this you can confirm that you have not forgotten any commas, or added too many, or missed any quotation marks: https://jsonlint.com/

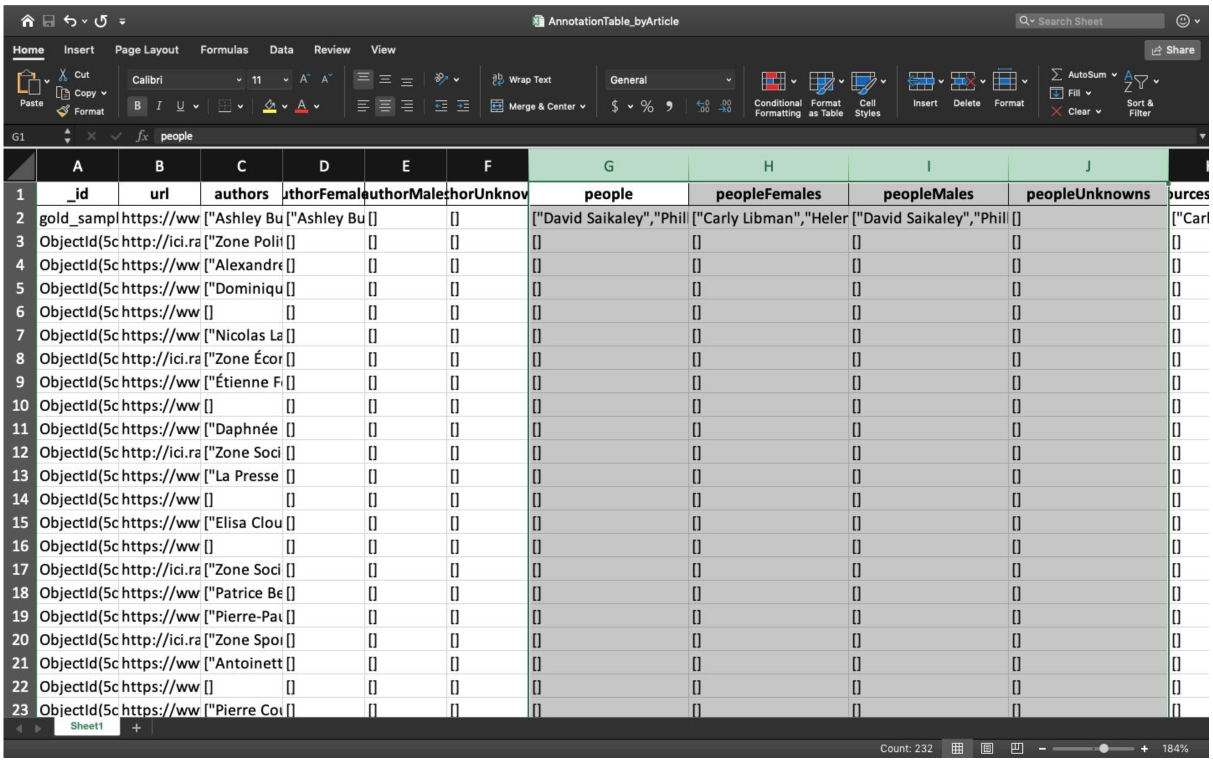
2) Annotating people for an article

(output: people columns in the "AnnotationTable_byArticle.xlsx" **file)**

The second set of fields that need to be annotated for each text are the people fields. Before you start the task, try to understand it by investigating the first data row in the "AnnotationTable_byArticle.xlsx" file, which contains the annotation of people fields for the gold sample. As you can see in the table, the **people** column contains an array of five people names. These are the full names of the people that have been mentioned in the sample article. Two of these names refer to female and three refer to male people. The gender of the person is determined based on the first names and/or the pronouns referring to them within the text. The column of **peopleFemales** contains a subset of items listed in **people** whose genders are resolved to female. The column of **peopleMales** contains a subset of items listed in **people** whose genders are resolved to male. If the gender of a person cannot be resolved based on the given criteria, they should be listed in the **peopleUnknowns** column.

For these people columns, we are only interested in cases where people are mentioned at least once in the text by their real name (full, first or last). So for instance, "a minister", "a German official", "A Toronto mom", etc., should not be considered in these fields if they are never mentioned by name. If they are quoted, their quotes can be included in the *quotations* field.

Important note: Please provide the names in all people fields in the full name format, unless the information cannot be obtained from the text (e.g., a person is always referred either by their first name or by last name only).



	A	B	C	D	E	F	G	H	I	J	
1	_id	url	authors	ıthorFemal	ıuthorMale	ıhorUnknov	people	peopleFemales	peopleMales	peopleUnknowns	ources
2	gold_sampl	https://ww	["Ashley Bu	["Ashley Bu	[]	[]	["David Saikaley","Phil	["Carly Libman","Heler	["David Saikaley","Phil	[]	["Carl
3	ObjectId(5c	http://ici.ra	["Zone Polit	[]	[]	[]	[]	[]	[]	[]	[]
4	ObjectId(5c	https://ww	["Alexandre	[]	[]	[]	[]	[]	[]	[]	[]
5	ObjectId(5c	https://ww	["Dominiqu	[]	[]	[]	[]	[]	[]	[]	[]
6	ObjectId(5c	https://ww	[]	[]	[]	[]	[]	[]	[]	[]	[]
7	ObjectId(5c	https://ww	["Nicolas La	[]	[]	[]	[]	[]	[]	[]	[]
8	ObjectId(5c	http://ici.ra	["Zone Écor	[]	[]	[]	[]	[]	[]	[]	[]
9	ObjectId(5c	https://ww	["Étienne F	[]	[]	[]	[]	[]	[]	[]	[]
10	ObjectId(5c	https://ww	[]	[]	[]	[]	[]	[]	[]	[]	[]
11	ObjectId(5c	https://ww	["Daphnée	[]	[]	[]	[]	[]	[]	[]	[]
12	ObjectId(5c	http://ici.ra	["Zone Soci	[]	[]	[]	[]	[]	[]	[]	[]
13	ObjectId(5c	https://ww	["La Presse	[]	[]	[]	[]	[]	[]	[]	[]
14	ObjectId(5c	https://ww	[]	[]	[]	[]	[]	[]	[]	[]	[]
15	ObjectId(5c	https://ww	["Elisa Clou	[]	[]	[]	[]	[]	[]	[]	[]
16	ObjectId(5c	https://ww	[]	[]	[]	[]	[]	[]	[]	[]	[]
17	ObjectId(5c	http://ici.ra	["Zone Soci	[]	[]	[]	[]	[]	[]	[]	[]
18	ObjectId(5c	https://ww	["Patrice Be	[]	[]	[]	[]	[]	[]	[]	[]
19	ObjectId(5c	https://ww	["Pierre-Pau	[]	[]	[]	[]	[]	[]	[]	[]
20	ObjectId(5c	http://ici.ra	["Zone Spoı	[]	[]	[]	[]	[]	[]	[]	[]
21	ObjectId(5c	https://ww	["Antoinett	[]	[]	[]	[]	[]	[]	[]	[]
22	ObjectId(5c	https://ww	[]	[]	[]	[]	[]	[]	[]	[]	[]
23	ObjectId(5c	https://ww	["Pierre Coı	[]	[]	[]	[]	[]	[]	[]	[]

Annotating the "people" fields.

3) Annotating sources for an article

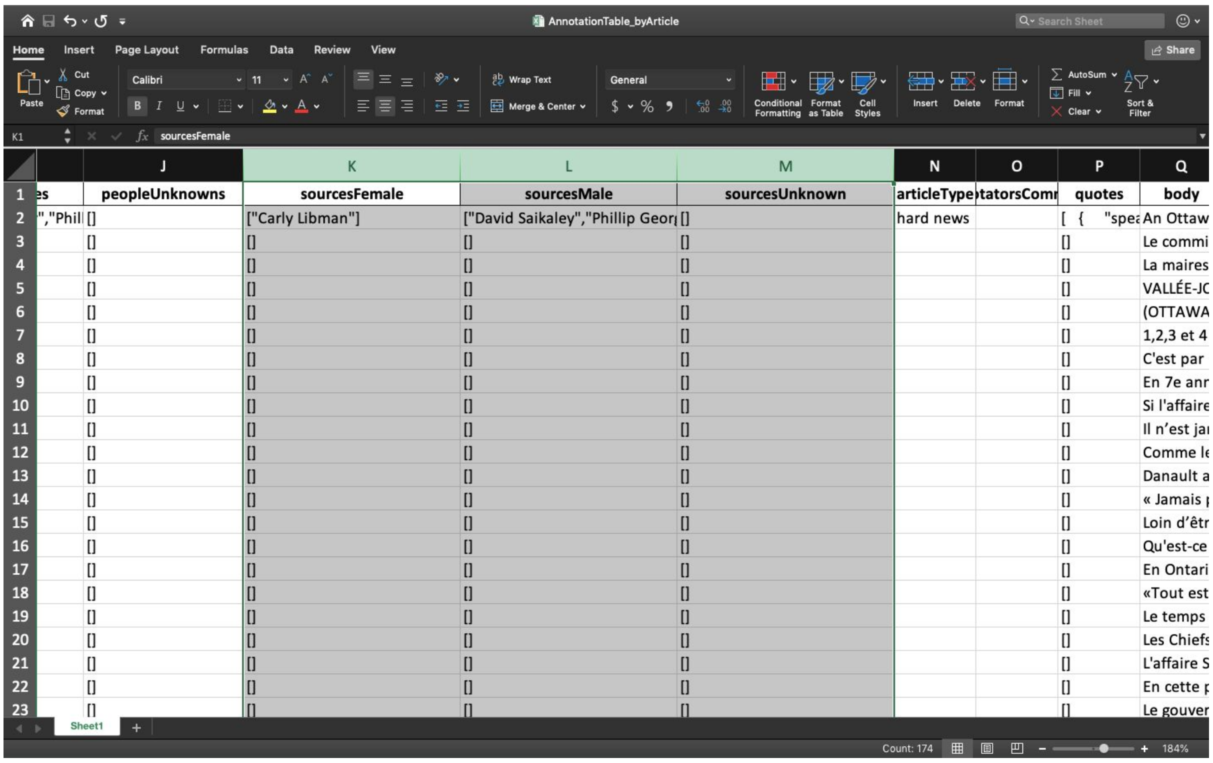
(output: source columns in the "AnnotationTable_byArticle.xlsx" **file)**

The third set of fields that need to be annotate for a text are the source fields. Before you start the task, try to understand it by investigating the first data row in the "AnnotationTable_byArticle.xlsx" file, which contains the annotation of source fields for the gold sample.

The column of **sourcesFemale** contains a subset of items listed in **peopleFemales** who were quoted somewhere in the text, i.e., played the speaker role in one of the quotations you found in annotation task 1. Similarly, the column of **sourcesMale** contains a subset of items listed in **peopleMales** who were quoted somewhere in the text, i.e., played the speaker role in one of the quotations you found in annotation task 1. Finally, **sourcesUnknown** contains a subset of **peopleUnknowns** who were quoted somewhere in the text, i.e., played the speaker role in one of the quotations you found in annotation task 1.

Important note: people who are quoted in the text are not necessarily mentioned by their full name where they are quoted. For example, the quotations associated with David Saikaley in the sample text include all cases where he was the speaker, even though in some places the pronoun "he", another type of reference such as "An Ottawa man acting as executor of a will" or even an empty string appears in the speaker role. You should count all these occurrences as quotations of David Saikaley.

Important note: not all speakers in the quotations within your JSON annotation file would end up in the people or source fields, because the speaker of some quotations might be organizations or they might be people not referred to by their names. For example, a few quotations by the TD bank do appear in the gold sample JSON file, but "TD bank" should not be included in source fields. Make sure that the subset relations between source and people fields always apply (e.g., **sourcesMale** should be a subset of **peopleMales**).



	es	peopleUnknowns	sourcesFemale	sourcesMale	sourcesUnknown	articleType	tatorsComr	quotes	body
2	","Phil	[]	["Carly Libman"]	["David Saikaley","Phillip Geor	[]	hard news		[{ "spea	An Ottaw
3		[]	[]	[]	[]			[]	Le commi
4		[]	[]	[]	[]			[]	La maires
5		[]	[]	[]	[]			[]	VALLÉE-JC
6		[]	[]	[]	[]			[]	(OTTAWA
7		[]	[]	[]	[]			[]	1,2,3 et 4
8		[]	[]	[]	[]			[]	C'est par
9		[]	[]	[]	[]			[]	En 7e anr
10		[]	[]	[]	[]			[]	Si l'affaire
11		[]	[]	[]	[]			[]	Il n'est jar
12		[]	[]	[]	[]			[]	Comme le
13		[]	[]	[]	[]			[]	Danault a
14		[]	[]	[]	[]			[]	« Jamais
15		[]	[]	[]	[]			[]	Loin d'êtr
16		[]	[]	[]	[]			[]	Qu'est-ce
17		[]	[]	[]	[]			[]	En Ontari
18		[]	[]	[]	[]			[]	«Tout est
19		[]	[]	[]	[]			[]	Le temps
20		[]	[]	[]	[]			[]	Les Chiefs
21		[]	[]	[]	[]			[]	L'affaire S
22		[]	[]	[]	[]			[]	En cette p
23		[]	[]	[]	[]			[]	Le gouver

Annotating the "source" fields.

4) Annotating the type of news article

(output: articleType column in the "AnnotationTable_byArticle.xlsx" **file)**

Finally, please provide one of the following values in the column of **articleType** to indicate whether the news article you read and annotated should be considered as a piece of "opinion" (the author's analysis of an event or phenomena) or "hard news" (reporting facts). If you are not sure about this decision, you can provide "undecided".

5) Comments

(output: annotatorsComment column in the "AnnotationTable_byArticle.xlsx" **file)**

During annotation of a news article, please use the annotatorsComment column within the AnnotationTable_byArticle.xlsx file for any comments or concerns.

Examples and unusual cases

Direct quote with speaker and verb

A direct quote involves repeating the exact words of the speaker. Typically these are surrounded by quotation marks, but they do not have to be.

Sentence from article:
```
Lorraine Hansberry said, "Never be afraid to sit awhile and think."
```

Annotation:
```
{
    "speaker": "Lorraine Hansberry",
    "verb": "said",
    "quote": "\"Never be afraid to sit awhile and think.\"",
    "speaker_index": "(21,41)",
    "verb_index": "(43,48)",
    "quote_index": "(51,85)",
    "reference": "Lorraine Hansberry"
}
```

Note the inclusion of the quotation marks and all the contents within it, including the final period. If you are confused by the backslashes that appear in the *quote* field, please read the introduction to JSON in the appendix.

Indirect quote / paraphrase

An indirect quote involves recounting the words of the speaker but they might be modified. Typically they are also recounted in third person, and they are not completely surrounded by quotation marks. Note that a paraphrase with a directly quoted word or phrase is possible, and discussed in the next example.

Sentence from article:
```
She alleged that we should never be afraid to sit awhile and think.
```

Annotation:
```
{
    "speaker": "She",
    "verb": "alleged",
    "quote": "that we should never be afraid to sit awhile and think",
    "speaker_index": "(21,26)",
    "verb_index": "(43,48)",
    "quote_index": "(51,85)",
    "reference": "Lorraine Hansberry"
}
```

Note that we do not include the period at the end as this is an indirect quote.

Paraphrase with direct quote within it

Sentence from article:

```
Lorraine Hansberry said that we should never be afraid to "sit awhile and
think."
```

Annotation:

```
{
    "speaker": "Lorraine Hansberry",
    "verb": "said",
    "quote": "that we should never be afraid to \"sit awhile and think.\"",
    "speaker_index": "(21,41)",
    "verb_index": "(43,48)",
    "quote_index": "(51,85)",
    "reference": "Lorraine Hansberry"
}
```

Note that since we want to preserve both quotation marks that appear within the sentence, we include the period at the end of the sentence.

Sentence from article:

```
Lorraine Hansberry explained that we should never be afraid to "sit awhile"
and think.
```

Annotation:

```
{
    "speaker": "Lorraine Hansberry",
    "verb": "explained",
    "quote": "that we should never be afraid to \"sit awhile\" and think",
    "speaker_index": "(21,41)",
    "verb_index": "(43,48)",
    "quote_index": "(51,85)",
    "reference": "Lorraine Hansberry"
}
```

Note that in this example we do not include the period.

Quote with no verb

Sometimes a quote may appear with a speaker but no verb. This is how it should be annotated.

Sentence from article:
```
We should never be afraid to sit awhile and think. - Lorraine Hansberry
```

Annotation:
```
{
    "speaker": "Lorraine Hansberry",
    "verb": "",
    "quote": "We should never be afraid to sit awhile and think.",
    "speaker_index": "(21,41)",
    "verb_index": "",
    "quote_index": "(51,85)",
    "reference": "Lorraine Hansberry"
}
```

Note that we include the period at the end as this is a direct quote. This is an example of a direct quote that appears without quotation marks. Also note that the *verb_index* field is completely empty.

Floating quote

When a quote has no speaker or verb, we call it a floating quote. These may or may not be surrounded by quotation marks, and most of the time the source of the quote can be inferred from context. Consider the second sentence in this text, which is an example of a floating quote.

Sentences from article:
```
"I had faulty advice," David Saikaley said. "I trusted the bank."
```

Annotation:
```
{
    "speaker": "",
    "verb": "",
    "quote": "\"I trusted the bank.\"",
    "speaker_index": "",
    "verb_index": "",
    "quote_index": "(82,103)",
    "reference": "David Saikaley"
}
```

Verbs with auxiliaries

An auxiliary verb like “to have” or “to be” is sometimes required to express a certain tense. Here is a future tense example: “He *will say* that he is tired”. This raises the question of whether “will say”, “will” or “say” are annotated as the verb. In all these cases, we would like only the main verb to be annotated, i.e., the one representing the action.

Sentence from article:

```
David Saikaley had been saying that he had faulty advice.
```

Annotation:

```
{
    "speaker": "David Saikaley",
    "verb": "saying",
    "quote": "that he had faulty advice",
    "speaker_index": "(0,23)",
    "verb_index": "(40,53)",
    "quote_index": "(55,72)",
    "reference": "David Saikaley"
}
```

Note that we leave out both auxiliaries - *had* and *been*.

Unusual quotative verbs

Not every quote is indicated with a verb related to speaking or saying things. Look out for more unusual verbs such as *urged*, *warned*, *complained*, etc.

Sentences from articles:

```
Rachel Notley warned not to allow the 22-week limit to drift.
Ms. Notley urged the passing of emergency legislation.
Saikaley complained that the bank that advised him wrongly was not helping
him.
```

Quote with an extra complement

Sometimes quotes have more information than just the speaker and what they said. They sometimes also include the person or organization being spoken to, or the manner in which the words were spoken. Please exclude of this information in your annotation.

Sentence from article:

```
She warned Ottawa not to allow the 22-week limit to drift.
```

Annotation:

```
{
    "speaker": "She",
    "verb": "warned",
    "quote": "not to allow the 22-week limit to drift",
    "speaker_index": "(0,23)",
    "verb_index": "(40,53)",
    "quote_index": "(55,72)",
    "reference": "Rachel Notley"
}
```

Note that we leave out “Ottawa”, the entity that she warns.

Sentence from article:

```
They said to TD Bank’s ombudsman, “Your service is wonderful!”
```

Annotation:

```
{
    "speaker": "They",
    "verb": "said",
    "quote": "“Your service is wonderful!”",
    "speaker_index": "(0,23)",
    "verb_index": "(40,53)",
    "quote_index": "(55,72)",
    "reference": "Oshawa family"
}
```

Note that we leave out “to TD Bank’s ombudsman”, the entity that they speak to.

Sentence from article:

```
He said firmly in a news conference this week that there was a lot of hope.
```

Annotation:

```
{
    "speaker": "He",
    "verb": "said",
    "quote": "that there was a lot of hope",
    "speaker_index": "(0,2)",
    "verb_index": "(4,7)",
    "quote_index": "(23,40)",
    "reference": "Barack Obama"
}
```

Note that we leave out “firmly”, the manner in which he spoke, and “in a news conference this week”, the location and time of his quote.

Quote with a possessive and noun

Sometimes a quote is introduced with a noun rather than a verb, e.g., “statement” instead of “stated”. Annotate that just like an example without a verb.

Sentence from article:

```
The instructor’s statement that the exam was cancelled met with cheers.
```

Annotation:

```
{
    "speaker": "The instructor",
    "verb": "",
    "quote": "that the exam was cancelled",
    "speaker_index": "(0,2)",
    "verb_index": "",
    "quote_index": "(23,40)",
    "reference": "Masha Cartwright"
}
```

Note that we leave out the possessive "'s", leaving just "The instructor" as the speaker. There is no verb, and we only use the part of the sentence that is the quote.

Questions and contact information

After studying the annotation guidelines and the gold sample, if you are confused about something please contact the person who sent you your annotation package. This also applies if you are confused about a particular unusual case of annotation you come across.

If you find an error in the gold sample, or a conflict between the annotation in the gold sample and the instructions in these guidelines, please notify us immediately. In case of inconsistencies, follow the guidelines in this document rather than following the gold sample.

You can also contact Fatemeh (ftorabia@sfu.ca) or Vasundhara (vgautam@sfu.ca) with any questions :)

Appendix I: Introduction to JSON

This section provides a more detailed introduction to JSON that you should read if you have never worked with JSON files before. Even if you are very comfortable with JSON, we advise you to skim through this section.

JSON (which standards for JavaScript Object Notation) is a format to store data that is easy for computers to work with. It is simple, very structured and easy to learn. For the purposes of this annotation task, we will focus on three things:

1. The overall structure of the JSON file you are supposed to write for each article
2. How to handle quotation marks when copying quotes from an article
3. How to check your file before sending it to us

Overall structure

A JSON file of quotations consists of a list of “quotation records” - one for each quote in the article. To tell the computer that it is a list of records, we need one pair of square brackets. All the quotation records will be listed within the square brackets and separated by commas.

If you were annotating an article that had no quotes in it, you would create a file with the following contents:

```
[
]
```

The square brackets indicate a list, but since there is nothing inside, it is an empty list. This may seem unnecessary but it is actually vital for our programs to be able to read and understand the data.

The format of a quotation record is as follows. There are 7 fields, all of which need to be mentioned, even if their contents are empty. Just like the quotation records are separated by commas within the square brackets of a list, the fields are separated by commas within the curly braces of a quotation record.

```
{
    "speaker": "...",
    "verb": "...",
    "quote": "...",
    "speaker_index": "(...,...)",
    "verb_index": "(...,...)",
    "quote_index": "(...,...)",
    "reference": "..."
}
```

From the previous sections, you should already be familiar with what each of those fields (*speaker*, *verb_index*, etc.) represents. In this section we will go into some of the lower-level details of how to fill these fields out in different situations.

In an ideal world you would have something to fill in every place where there are dots in the example above. A fully filled out example would look like this:

```
{
    "speaker": "David Saikaley",
    "verb": "worries",
    "quote": "the family members may never see their inheritance",
    "speaker_index": "(306,320)",
    "verb_index": "(321,328)",
    "quote_index": "(329,379)",
    "reference": "David Saikaley"
}
```

There will alway be something in the *quote* field, but there may not always be something in the *speaker* or *verb* fields. It is important to note that if any of the fields is not there, you must still have the quotation marks on the outside, just the inside will be empty:

```
✓ "speaker": "David Saikaley",
✓ "speaker": "",
❌ "speaker": ,
```

In the case of no speaker, the *speaker_index* field should also look identical to the empty *speaker* field, i.e, the quotation marks should stay but there should be nothing within them.

```
✓ "speaker_index": "(306,320)",
✓ "speaker_index": "",
❌ "speaker_index": "(,)",
❌ "speaker_index": "()",
❌ "speaker_index": ,
```

Here is a short example of the contents of a correct and complete JSON file:

```
[
  {
    "speaker": "David Saikaley",
    "verb": "worries",
    "quote": "the family members may never see their inheritance",
    "speaker_index": "(306,320)",
    "verb_index": "(321,328)",
    "quote_index": "(329,379)",
    "reference": "David Saikaley"
  },
  {
    "speaker": "",
    "verb": "",
    "quote": "\"I would have sought out a different method for sending the
money.\"",
    "speaker_index": "",
    "verb_index": "",
    "quote_index": "(4992,5059)",
    "reference": "David Saikaley"
  }
]
```

Notice that after the final field in each quotation record (the *reference* field), there is no comma. Similarly after the last quotation record in the list, there is no comma either.

Handling quotation marks

As you have seen in the previous subsection, quotation marks are part of the structure of a JSON file. When the first quotation mark opens, the computer knows that it has to pay attention to whatever is coming next, and when it closes, the computer knows that there is no more important information like a field name or the name of a speaker.

For example, when it sees “speaker” then it expects that the next letter it will see is a colon (:) character. When it sees the : character, it expects another pair of quotes containing the speaker’s name, if there is a speaker. After that it expects a comma, to show that it is the end of that field, and so on.

So when we have quotation marks in our quotations that we want to paste into the *quote* field, this creates problems. Imagine the computer trying to read the following line:

❌ `"quote": ""Hello!"",`

The computer sees the field name for the *quote* field, (“quote”), and it sees a colon. All is going as expected so far. It sees one quote character (“) and starts to pay attention, to expect the content of the quote. But it doesn’t see alphabets yet; it sees another quote character (“)! Now it assumes that this means there is no quote here, and it expects a comma next, and then the next field. But after that, it doesn’t get a comma! It gets an “H”. It assumes the data is wrong or broken and so it stops working. Obviously this is not something we want, so how do we tell the computer that “Hello!” is the content of the quote?

Fortunately, the creators of JSON foresaw this problem and gave us ways to tell the computer to ignore a quotation mark inside quotation marks that is relevant. If you want a quotation mark to be ignored, just put a backslash (\) in front of it! The right way to annotate the example above is as follows:

```
✔️ "quote": "\"Hello!\"",
```

Now after seeing the colon and the first quotation mark, the computer sees a backslash. This tells the computer to ignore the special meaning of whatever the next character is. So when it sees another quotation mark after that, it just ignores its “special” meaning and proceeds. After it sees the exclamation mark (!), it sees a backslash again. As before, it knows to just ignore the special meaning of the next quotation mark and proceeds. After this it sees another quotation mark (the 4th and last one), but there was no backslash before it, so it knows to take this one seriously. It understands that the quote is over, and then it expects a comma after, which it gets.

Other quotation marks

We have found in some articles that different publications represent quotation marks in different ways. The key thing to remember is that you only need to pay attention this issue of quotation marks, if the quotation marks in the JSON file structure are identical to the ones used in the article. The computer only gets confused if there is a special meaning assigned to whatever it is seeing.

With quotation marks like this (“) in your JSON files, but quotes in the article are represented with French guillemets («) then you do not have to put a backslash in front of the guillemets. Here are some examples:

```
❌ "quote": ""Hello!"",
✔️ "quote": "\"Hello!\"",

❌ "quote": "\«Bonjour!\»",
✔️ "quote": "«Bonjour!»",

❌ "quote": "\’Hello!\’",
✔️ "quote": "’Hello!’",

❌ "quote": "\”Hello!\”",
✔️ "quote": "”Hello!”",
```

The one that is probably the most confusing is the last one, where all the characters look like double quotes. However, if you look closely you will see that the outside quotation marks are “straighter” than the inner ones, which are slanted. Computers use the straight quotes to understand JSON files, and they don’t care about slanted quotes. This is why you do not need to put a backslash in front of those.

Checking your file

By now you have probably realized that there are many very minor things that could go wrong in a file and cause our programs trouble. As humans we would be able to read and understand a file even if it did not have backslashes or such a strict format with commas and quotes, but computers are not that smart or flexible.

Luckily there are plenty of tools available online called JSON validators. Their job is to scan through your file and point to the exact location of your errors, if any. These are very helpful during annotation and it is important that you use them to check that your JSON files are valid before sending them to us.

Here are a few you can try out. Remember that you need to paste a fully valid file, i.e., including square brackets, curly braces and all the fields. Just one line like some of the examples above will not work.

https://jsonformatter.org/
https://jsonlint.com/
https://jsonformatter.curiousconcept.com/