University of Alberta
CMPUT 497/501 Fall 2019
Assignment 1

Due September 25 2019, on eClass

Learning Objectives.

Information Extraction (IE) is the task of extracting and organizing data (e.g., facts about entities) expressed in text into a query-able and structured knowledge base.

Browse https://web.stanford.edu/~jurafsky/slp3/17.pdf for an introduction to the task.

For the purpose of this assignment, we will be concerned with extracting facts that are explicitly stated in Wikipedia articles. That is, you are not expected to extract facts that are either implied or that are supported only by multiple articles (or even multiple sentences in the same article).

For example, the text "Finding Nemo is a 2003 American computer-animated adventure film produced by Pixar Animation Studios and released by Walt Disney Pictures." expresses many explicit facts that are amenable to IE systems. We will represent them here as triples:

```
(Finding_Nemo, type, Film)
(Finding_Nemo, year, 2003)
(Finding_Nemo, producer, Pixar_Animation_Studios)
...
```

The objective of this assignment is for you to learn how to build and evaluate a reasonable IE tool for Wikipedia articles that, like earlier systems in this area, relies primarily on regular expressions that exploit patterns in the text.

The input: Wikipedia Articles

Wikipedia is a vast collection of hyperlinked *semi-structured* articles written by humans and exploited for many knowledge management and NLP purposes.

Here's an excerpt from the Wikipedia article about Finding Nemo:

```
{{about|the film|the franchise|Finding Nemo (franchise)|the video game|Finding Nemo (video
game)}}
{{good article}}
```

```
{{Use mdy dates|date=January 2018}}
{{Infobox film
| name           = Finding Nemo
...
| director       = [[Andrew Stanton]]
…
'''''Finding Nemo''''' is a 2003 American<!-- Please do NOT change it to Australian. Despite
being set in Australia, the film was developed and produced by an American studio.-->
[[computer animation|computer-animated]] [[adventure film]] produced by [[Pixar Animation
Studios]] …
```

Most Wikipedia articles open with metadata, enclosed by {{ and }}. One particularly useful piece of metadata often used by IE systems on Wikipedia are the “InfoBoxes” which are visible to the reader as a side table with a summary of the entity (or event) that is the main subject of the article.

Some of the information expressed in the InfoBox is repeated in the body of the article. For example, in the case of Finding Nemo, both the text of the article and the InfoBox indicate that Pixar was the studio behind the movie.

The output: TSV file with triples and an attribution

The output of your program should be a TSV file with 4 columns:

- Subject
- Predicate
- Object
- Evidence

Each (subject, predicate, object) triple should express a fact that you extracted from the article (as in the examples above). The evidence field should be the **shortest** span of text that supports the fact. For example, the fact `(Finding_Nemo, type, Film)` is supported by the text “Finding Nemo is a 2003 American computer-animated adventure film”. We will use the evidence field to judge your output. Facts extracted without convincing evidence will not be considered.

You are expected to extract triples from the Infoboxes and from the text. The support for a fact from the infobox can be the line of the infobox with that fact. You are also expected to avoid duplicating facts, regardless of where they were extracted from.

Reference set of triples:

We provide a minimalistic set of facts for each of the movies for reference only. Those facts come from the DBpedia project, which employs a number of simple IE tools (many of which rely on regular expressions over InfoBoxes only) to derive a Knowledge Base from Wikipedia. You should use those facts to guide your work, instead of as the final goal.

Suggestions on how to proceed.

General suggestions:

- If/when in doubt, ask for help right away. This is an open-ended assignment.
- Familiarize yourself with Wikipedia's markup format. Exploit its regularity to identify entities (most of which are proper nouns) and the phrases used to express relations.
- Consider resolving mentions to entities by their name (e.g., entities mentioned in the InfoBox appearing in the body of the text)
- Look for relationships among entities that are not in any InfoBoxes.

Suggestions about code structure:

- Split the text into sentences; this will help you focus your search on smaller text samples. If you use a library for this step (e.g., NLTK), pay attention to what is done. Keep in mind that you don't have to use any third-party library.
- Focus on a set of facts to retrieve at at time. Do not tackle many relations at once.
- Create regular expressions to look for each fact in your set. Each regular expression match must extract the subject, predicate, and object from the sentence.
- You should expect that there will be many regular expressions for each relation.
- It is possible that the word used in the sentence does not exactly match the fact keyword, e.g., the fact *producer* can be extracted from a sentence with the format "subject *produced by* object"
- Add comments after regular expressions definition indicating to which fact they extract.
- Consider a nested for-loop structure in which you test every sentence with every regular expression until you find matches.
- If you find the same fact twice (e.g., from different sentences), it will be up to you to decide which one to keep.
- One of the challenges you might run into is that there are multiple ways to express any given relation.
  - For instance, to extract facts about the film's cast, you may need to look for variations of the sentence "Subject *stars* object" and write regular expressions to match them. The following excerpts may help you understand the challenge:
    - "the film stars the voices of Albert Brooks, Ellen DeGeneres, Alexander Gould, and Willem Dafoe."
    - "The film features an ensemble voice cast that includes Aki Asakura, Kengo Kora, Takeo Chii, ..."
    - "The film's voice cast features Kelly Macdonald, Billy Connolly, Emma Thompson, ...".

Grading rubric:

50% : Code Clarity (how easily can the TA understand your solution).

perc.	description
100	Clear execution instructions are provided. TA can understand the code without effort, including the precedence of the regular expressions, and the execution instructions are clear and correct. Every fact is extracted by a single regular expression, and all regular expressions are documented. Duplicated facts are removed in a clear and systematic way.
75	The TA can understand the code without effort but the execution instructions are unclear, incorrect or missing.
50	The TA can understand the code but the facts are extracted using a combination of regular expressions and other coding constructs in the code. OR the regular expressions are not clearly documented.
25	The code is not broken down into concise and properly named functions; the TA needs to make several assumptions and leaps to get a sense of what the code does.
0	he TA cannot execute the code with the instructions provided despite reasonable efforts; OR the TA cannot understand the code despite reasonable efforts. OR the code produces only hard-coded facts without regular expressions.

30% Precision (how convincing are your regular expressions).

perc.	description
100	All facts are properly supported; the evidence is short and readable text.
75	All facts are supported, but the evidence is vague, too long (e.g., entire sentences), or unreadable.
50	All facts seem correct but not all of them are not supported by the evidence.
25	Many of the facts seem incorrect.
0	None of the facts are supported by the evidence provided. OR the code produces only hard-coded facts without regular expressions.

20% Recall (how many good facts your regular expressions extract).

perc.	description
100	The code extracts facts from the text and from the infoboxes; the code covers all relations in the sample set and a few more. All facts in the sample set are extracted.
75	The code extracts facts from the text and from the infoboxes; the code covers all relations in the sample set and a few more.
50	The code extracts facts from the text OR from the infoboxes but not both. OR the code covers fewer relations than in the sample.
25	The code extracts fewer facts than in the sample set.
0	The code produces only hard-coded facts without regular expressions.