University of Alberta
CMPUT 497/501 Fall 2019
Assignment 4

Deadline and submission instructions: check eClass.

Learning Objectives.

Understanding distant supervision, a popular method for obtaining large quantities of training data for NLP tasks, and some of its limitations with respect to the quality of such training data.

Distant Supervision.

One trade-off when using supervised learning algorithms concerns the number of training examples. On one hand we want a large number of examples to improve performance, but on the other we want to minimize the cost of obtaining them. This is a serious issue that has attracted a lot of attention from researchers. One way to address it, called distant supervision, was proposed for the NLP task of relation extraction from text, illustrated by the examples below (where the relation in question is the **genre** of a film):

> S1: “Directed by Jack Nicholson himself, Goin’ South is a spotty Western comedy that offers modest returns.”
>
> S2: “George Clooney is ready to take on a dark spy comedy called Burn after Reading.”

Note that S1 explicitly states that “Goin’ South” is a “Western comedy” through the “is a” phrase, while S2 implies that “Burn after Reading” is a “comedy” because of “called”. Both are good examples for a classifier. Now, consider:

> S3: “Fittingly, a mere three years later, Yojimbo was remade for more widespread Western consumption by Sergio Leone as A Fistful of Dollars, the movie that introduced the world to the Spaghetti Western and made Clint Eastwood into a star.”
>
> S4: “Clooney also starred in the Coen brothers’ movie O Brother, Where Art Thou, for which he won the 2000 Golden Globe Award as Best Actor in a Motion Picture Musical or Comedy.”

S3 establishes that “A Fistful of Dollars” is a “Spaghetti Western”, through a much less direct way, although it could still be used as an example of the film genre relation. S4, on the other hand, would be a bad example as it does not state the genre of “O Brother, Where Art Thou”.

It should not be hard to see that having humans inspect and label a large corpus of sentences like those (all of which were extracted from the Web) to decide which are good and which are bad training examples would incur a high cost. The text comprehension skills required of the annotators alone render their time too expensive for large-scale annotations exercises.

Distant Supervision, introduced by Mintz. et al. in 2009, suggests a method for obtaining a large number of *silver standard* annotations (as opposed to gold standard annotations provided by humans). They start from a large Knowledge Graph that knows about many kinds of entities such as movies and movie genres. Next, they use Named Entity Recognition (Chapter 18 of the Jurafsky Martin book) tools to find mentions to entities in a large corpus (e.g., from a Web crawl) and disambiguate such mentions to the entities in the text with identifiers from the Knowledge graph. Finally, they query the knowledge graph to find many pairs of entities that belong to the relation of interest (e.g., individual movies and their genres) and use all sentences tagged with those entities as training data for that relation.

Here is an example of what the annotated text looks like, using Freebase IDs for the entities:

```
Made in 1966 - the same year in which [[ Corbucci | /m/08ng1r ]] directed both [[ Django | /m/09n5sx ]] and [[ Navajo Joe | /m/09znl2 ]] - The [[ Hellbenders | /m/0l3pnc ]] again reveals [[ Sergio Corbucci | /m/08ng1r ]] to be a filmmaker of exceptional talent, armed with a keen understanding and admiration for the mystique of the [[ Western | /m/0hfjk ]].
```

Noise introduced by distant supervision.

In order to scale to large corpora, distant supervision requires the use of crude and fast NLP methods, which inevitably make mistakes leading to noisy training data. In this assignment we will consider only two kinds of such problems.

(1) Words incorrectly identified as mentions to entities, as in:

```
The fact that this is made by [[ Bioware | /m/0j2c3 ]] as well also gives me some warm pleasure seeing how [[ Microsoft | /m/04sv4 ]] is [[ Publishing | /m/0hz28 ]] the title.
```

Note that the verb “publishing” is incorrectly tagged with the identifier of the *business segment* of book printing and distribution, to which Microsoft actually belongs. Therefore, that sentence should not be taken as a training example for the business segment relation.

(2) Entities involved in the relationship detached from one another in the text, as in:

```
The judges for the Richmond Regional Final included [[ David Wojahn | /m/027cxs8 ]] , Director of the [[ Creative Writing Program | /m/040p_q ]] , [[ Virginia Commonwealth University | /m/0177sq ]] ; Bruce Miller, Artistic Director of Theater IV/Barksdale; [[ Natasha Tretheway | /m/02qn51h ]] , [[ Poet | /m/05z96 ]] and Writer in Residence at Duke Center for Documentary Studies; and Mary Flinn, Director of the New Virginia Review.
```

Note that the sentence mentions a person, David Wojahn, and their *profession*, poet, although there is no connection between these entities in the sentence.

The tasks.

In this assignment you are given several JSON files, one per relation, with hundreds of annotated sentences extracted using distant supervision, and your task is to implement filters for detecting problematic sentences which should not be used as training examples.

TASK 1 (25%): POS tagging to detect mislabelled entities.

Your first task is to write a program to identify words or phrases tagged with entity identifiers in the data but which are unlikely to be nouns (and therefore entities) according to a probabilistic POS tagger of your choice.

Everything in this part should be submitted in a folder called `task1`.

Evaluation. You will be evaluated based on correctness and clarity of the code you submit (40%) and your README file giving instructions on how to run it (10%), a report about your findings (40%) and the output of your program (10%).

Report. You are asked to submit a concise report with your findings. The report must contain a table showing, per relation, estimates of the number of filtered sentences and the average number of misidentified entities per sentence. To find these statistics, sample 100 sentences from each relation and manually inspect the output of your program on them. Your report should also answer the following questions. Are there identifiable patterns related to filtered sentences? Which are the most common POS tags associated with misidentified entities?

Output. In a folder called `runs` inside `task1` put one plain text file for each given relation containing: an original sentence in a single line, followed by the list of POS tags for that sentence (one word per line) and any entities that your method deemed to be incorrectly annotated (one per line). Add two blank lines before the next sentence.

TASK 2 (75%): Filtering out spurious relational sentences.

Your second task is to write a program to navigate a dependency tree of a sentence to determine if the relationship in question holds between the given pair of entities. For simplicity, we ask that you focus on relationships expressed through verbs. You are expected to judge which verbs are suitable for each relation (is “play” a good choice for the actor-role relation?).

Everything in this part should be submitted in a folder called `task2`.

Evaluation. You will be evaluated based on correctness and clarity of the code you submit (40%) and your README file giving instructions on how to run it (10%), a report about your findings (40%) and the output of your program (10%).

Pre-processing. Attempting to obtain reliable dependencies among words from sentences with entity annotations as in the files provided is a bad idea because the annotations are more than likely to confuse the parser. To avoid this problem you are **required** to replace mentions to entities as follows: the entities identified as the subject and the object in the sentence should be replaced by SUBJECT and OBJECT, respectively. All other entities should be replaced by ENTITY*j* where *j* is a counter, as in the example:

```
Lost in Translation: In a career-best performance, SUBJECT plays OBJECT, an ENTITY1 movie
star who arrives in ENTITY2 to film a series of television commercials.
```

Where

```
"SUBJECT": "[[ Bill Murray | /m/0p_pd ]]"
"OBJECT": "[[ Bob Harris | /m/02nw8hl  ]]"
"ENTITY1": "[[ American | /m/09c7w0 ]]"
"ENTITY2": "[[ Tokyo | /m/07dfk ]]"
```

Paths between subject and object. The goal of your program is to find the node in the tree that is the *lowest common ancestor* connecting SUBJECT and OBJECT, if at all possible. In the example above, that node corresponds to the word “plays”. To do so, we recommend you navigate the tree bottom-up to find the paths from each node to the root and then intersect such paths. We strongly recommend you use Spacy.IO for this assignment as it offers good support for navigating dependencies.

Report. You are asked to submit a concise report with your findings. The report must contain a table showing, per relation, estimates of the number of sentences deemed to actually express the relation, and the average number of different verb stems in them. To find these statistics, sample 100 sentences from each relation and manually inspect the output of your program on them. You are encouraged to consider relations expressed nonverbally as well. If you do so, add other columns to your table to include relevant statistics. Finally, discuss in your report which verbs are more frequently used to mediate each relation, and the forms in which they are used (e.g., active vs passive voice, present or past tense, etc.). Are there verbs that *do not express* the desired relation but nevertheless appear in the data? What about relations mediated by “is”?

Output. In a folder called `runs` inside `task2` put one plain text file for each given relation containing: the sentence with entities replaced in one line, followed by the mappings between actual entities and their surrogates (one per line), followed by the paths from the SUBJECT and OBJECT to the root of the tree (as many lines as needed to make the output readable), and the lowest common ancestor. Add two blank lines before the next sentence.