CS-5340/6340 PROJECT DESCRIPTION, Fall 2019

The project for this class will be to design and build your own coreference resolution system. For each document, we will give you the **initial reference** for every coreference cluster. Your job will be to find all other references that belong in the same cluster.

You are free to build the system however you want! We will specify what the input and output should be, but otherwise the design of the system is up to you. We recommend that you work in a 2-person team for the project. But if you strongly prefer to work alone, then you can work as a solo team.

Your grade for the project will be based on how well your coreference resolver performs. We will provide gold standard coreference data to help you understand the task and develop your systems. We will also provide a scoring program that you can use to evaluate the performance of your system.

INPUT FILE SPECIFICATIONS

Each document will be marked up with two types of information: (1) sentence boundaries, and (2) the initial reference for every coreference cluster.

Each sentence will be surrounded by the tags <S ID="#"> and </S>, where ID indicates the sentence number (#).

Each initial reference will be surrounded by the tags <COREF ID="X#"> and </COREF>, where the ID is a unique identifier (X#) for the reference. The initial reference will be the earliest mention of the cluster concept in the document. You can assume that there will be at least one additional reference belonging to each cluster, that all additional references will occur after the initial reference, and that each reference will be a pronoun, noun, or noun phrase. You do NOT need to find possessive references (e.g., "his" or "Susan's").

As an example, a short story might look like this:

```
<S ID="0"> <COREF ID="X0">Susan Mills</COREF> bought <COREF ID="X1">a home</COREF> in Utah.</S>
<S ID="1">A nice feature is that the 2-story house has a big yard for <COREF ID="X2">her dog</COREF>.</S>
<S ID="2">The German Shepherd weighs 100 lbs and is very active.</S>
<S ID="3">Both Sue and the dog love the new house!</S>
```

This story has 4 sentences and 3 coreference clusters. The initial reference for the first cluster is "Susan Mills", the initial reference for the second cluster is "a home", and the initial reference for the third cluster is "her dog".

ANSWER KEY (GOLD STANDARD) FILE

We will provide you with answer key files for the development set, which will contain the gold standard coreference clusters for each document. You can use the answer key files to better understand the nature of the task, to evaluate the performance of your system on the development set stories, and to train a machine learning system (if you wish to use ML for your project).

Each coreference cluster in the answer key files will begin with the initial reference (as marked up in the document) followed by each coreferent phrase in the document on a separate line. Each coreferent phrase will be indicated by the *sentence number* where it occurs, its *maximal phrase*, and its *minimal phrase*. The maximal phrase is the full text span for the coreferent phrase, and the minimal phrase is the shortest acceptable text span for the coreferent phrase, which is usually the head noun or a named entity span. The specific format is shown below:

```
INITIAL REFERENCE
{SENTENCE_ID} {MAXIMAL_PHRASE} {MINIMAL_PHRASE}
{SENTENCE_ID} {MAXIMAL_PHRASE} {MINIMAL_PHRASE}
etc.
```

Each cluster's information will be separated by a blank line. For the sample document shown earlier, the answer key file would look like this:

```
<COREF ID="X0">Susan Mills</COREF>
{3} {Sue} {Sue}

<COREF ID="X1">a home</COREF>
{1} {the 2-story house} {house}
{3} {the new house} {house}

<COREF ID="X2">her dog</COREF>
{2} {The German Shepherd} {German Shepherd}
{3} {the dog} {dog}
```

Your coreference system will be scored based on its ability to find the correct coreferent phrases for each cluster, and not to produce spurious resolutions. A system's response phrase will be considered correct if the phrase is in the correct sentence, includes the minimal span, and does not exceed the maximal span.

For example, consider the coreference cluster for "a home". Any of the following phrases from Sentence 1 will be considered correct: "the 2-story house", "2-story house", and "house". But these phrases would be considered incorrect: "that the 2-story house" (exceeds maximal span), "house has" (exceeds maximal span), and "the house" (because the phrase is not a contiguous span in the document).

OUTPUT (RESPONSE) FILE SPECIFICATIONS

Your coreference system should produce a *response file* for each document that lists the coreference phrases found for each cluster. The format will be similar to the answer key files, except that you should produce just a single phrase for each answer (whereas the answer key has maximal and minimal spans). The specific format is shown below:

```
INITIAL REFERENCE
{SENTENCE_ID} {ANSWER_PHRASE}
{SENTENCE_ID} {ANSWER_PHRASE}
etc.
```

The information for each cluster will be separated by a blank line. For example, a response file for the previous story might look like this:

```
<COREF ID="X0">Susan Mills</COREF>
{1} {A nice feature}
{3} {Sue}

<COREF ID="X1">a home</COREF>
{1} {house has}

<COREF ID="X2">her dog</COREF>
{3} {dog}
```

If your system produced this response file, two of the answers would be considered correct ("Sue" and "dog") and two of the answers would be considered incorrect ("A nice feature" and "house has"). The system's recall would be 2/5 because it found 2 of the 5 coreferent phrases in the answer key, and its precision would be 2/4 because 2 of the 4 answers that it generated are correct.

Please list the mentions in the order that they occur in the document, with the earliest mention first. This is especially important when there are multiple referents in the same sentence. For example, consider the sentence: *John Smith is a singer and John has won many awards.* Be sure to list "John Smith" in your list of referents before "John".

I/O SPECIFICATIONS

Your coreference resolver should accept two arguments as input:

1. A **ListFile** that specifies the full pathname for the input files to be processed, one per line. Each input file should be named "StoryID.input". For example, a ListFile might look like this:

   ```
   /home/yourname/project/coref/data/1.input
   /home/yourname/project/coref/data/52.input
   /home/yourname/project/coref/data/33.input
   ```

2. A directory name (string), where your program's output (response files) will be written.

For example, if you are using python3 then we should be able to run your program like this:

```
python3 coref <ListFile> <ResponseDir>
```

and we might run your program this way:

```
python3 coref test1.listfile /home/yourname/project/coref/output/
```

For each input file, your coreference resolver should produce a new file with the same prefix but the extension **.response** instead of **.input**. For example, given 52.input, your program should generate an output file named 52.response. All of the response files should be put in the directory specified on the command line.

RESOURCES

We will give you three types of data to use while developing your systems:

1. **Input files:** the input files for the development set documents, which will end with a **.input** extension.
2. **Answer key files:** the gold standard answer keys for each story in the development set (i.e., the correct answers that your program's output will be scored against). These files will end with a **.key** extension. These files must be provided to the scoring program in order to compute the accuracy of your coreference resolver's output.
3. **Scoring Program:** a python program that you can use to score your coreference resolver yourself! This is exactly the same program that we will use to evaluate your systems. By giving you a copy of our scoring program, you can evaluate the performance of your coreference resolver as often as you like, for example to try out new ideas and test new components.

All of these files will be available in the Project folder on Canvas.

EXTERNAL SOFTWARE

You may use external software packages for your project, as long as the following criteria are met:

- **You may NOT use any external software that performs any type of coreference resolution whatsoever.** This includes pronoun resolvers, anaphora resolution, or any software that performs resolutions of NPs or other syntactic constituents. You also may not use any external software that performs functions specifically related to coreference resolution, such as non-anaphoric NP identification. If we discover that your submitted system uses any external coreference system or code, you will get a zero for the project.
- You must fully acknowledge in your final presentations/posters all of the external software that you used in your project system.
- You must be able to run the software on the linux-based CADE machines and allow us to be able to run it as well during grading. This means either including the software in your code submission, or installing it in your own CADE directory and giving us full permission to access it from there.

You are allowed to use external NLP software that performs other basic NLP functions, such as tokenizers, part-of-speech taggers, general-purpose syntactic parsers (as long as they do not perform coreference resolution), and general-purpose semantic dictionaries such as WordNet.

If you are uncertain about whether a specific tool is acceptable, please ask us!

Machine Learning

You do **NOT** need to use machine learning (ML) for this project. But you are welcome to do so if you wish. If you choose to use ML:

- You MAY use external ML software packages, such as scikit-learn.
- You may NOT use any ML models that have been previously trained for coreference resolution or a subtask related to coreference resolution, such as non-anaphoric NP identification. If you create an ML model for coreference resolution, you must train it yourself using ONLY the cs5340/6340 data.
- You MAY use the Development Set as training data for both the Midpoint and Final Evaluations. You may use Test Set #1 as additional training data for the Final Evaluation, if you wish.
- You may NOT use any additional sources of training data. If you submit an ML model that has been trained on any external data, your coreference system will be disqualified and you will get a zero for the project. The reason is that we want to level the playing field, so that everyone is using exactly the same data set to build their systems.

EVALUATIONS

The project will involve three phases:

1) Development Phase: You will be given a **Development Data Set** to use when designing your coreference resolver. You may use this data in any way that you wish. I strongly encourage everyone to spend time reading the documents and the answer keys. You can also use this data to train a machine learning classifier if you want.

2) Midpoint Evaluation: For the midpoint evaluation, each team's system will be evaluated on a brand new set of documents, which we will call **Test Set #1**. Grades will be assigned based on each team's performance on Test Set #1 relative to the other teams. The purpose of this evaluation is to make sure that every team is making progress on creating a coreference resolver and to allow everyone to see how other teams are performing at the (roughly) halfway point.

30% of your overall project grade will be based on your system's performance on Test Set #1. Once the midpoint evaluation is over, we will release Test Set #1.

4) Final Evaluation: For the final evaluation, we will run each system on a second test set: **Test Set #2**. 65% of your overall project grade will be based on your system's performance on Test Set #2.

Scoring

The performance of your coreference resolver will be based on its **F score**, which is a harmonic mean of Recall and Precision. These scores are defined as:

Recall (R): the number of correct references identified by your system divided by the total number of references in the answer key.

Precision (P): the number of correct references identified by your system divided by the total number of references produced by your system.

F Score: F(R,P) $=\frac{2 \times P \times R}{P + R}$
This formula measures the balance between recall and precision (it is the harmonic mean). The final performance of each system will be based on its F Score.

SCHEDULE

Wednesday, Oct. 16 by 11:59pm: fill out and submit the Team Request Form in the Project folder on Canvas.

Monday, Nov. 4: Midpoint Evaluation on Test Set #1. The documents and answer keys for Test Set #1 will be released after the scoring is completed.

Monday, Nov. 25 (by NOON!): Final System Evaluation on Test Set #2.

Monday, Dec. 2: In-class oral presentations by the top 5 teams.

Wednesday, Dec. 4: In-class poster session.

Wednesday, Dec. 4: Project presentation slides & posters due.

Important: No late submissions will be allowed for any aspects of the project! Submissions will not be accepted after the due dates above.

During the in-class presentations and poster session, each team will describe how their system works and discuss how their system performed on the test sets. Details on the project presentations and posters will be forthcoming.

GRADING

Each project will be graded according to the following criteria:

- 30% of the grade will be based on the performance of your coreference resolver on Test Set #1 during the Midpoint Evaluation.
- 65% of the grade will be based on the performance of your coreference resolver on Test Set #2 during the Final Evaluation.
- 5% of the grade will be based on your project presentation, which will be an in-class presentation for the 5 top-performing teams or a poster presentation for the remaining teams. All teams will also be required to submit their slides or posters for grading.

To compute these grades, all of the teams will be ranked based on their performance. Clusters will be created for teams with systems that achieved similar scores, and then grades will be assigned based on the ranked clusters. The highest-scoring cluster will get the top grade, the second highest-scoring cluster will get the second top grade, etc.

The final grading will primarily be based on how well your system does relative to other teams' systems, but it is not the case that the highest ranked system will automatically get an 'A' or that the lowest ranked system will automatically get an 'E'. If every team produces a system that works wonderfully well, then I will be thrilled to give everyone an 'A' on their project! If, at the other extreme, no team generates a system that works at all, then I would be forced to give every team a failing grade. I hope, and expect, that everyone will create an effective system and get in the competitive spirit to try to build the best system you can!

It is fine to share ideas with other teams (but not code!), and to compare your system's performance with other teams. But if your team is doing almost exactly the same thing as many other teams, chances are your system will end up in the middle of the rankings. To distinguish yourself and stand apart in the rankings, we encourage teams to try different things!

Important: For the project presentations/posters, I will ask you to clearly delineate *your individual* contribution to the project, in terms of which software components you developed and how much overall effort you contributed to the project. This will allow me to adjust individual grades in case some people put in much more effort than others. Every team member must take responsibility for some software components in the project! If one teammate contributes very little to the project, then that person will get a lower grade than the person who put in substantially more time and effort.

Caveat: Coreference resolution is hard! Coreference resolution is an open research problem; even state-of-the-art coreference resolvers still achieve only mediocre performance. Your task is simplifed from the full coreference problem, but even so, I do not expect your systems to get extremely high scores. Your goal is to build the best coreference resolver that you can.

I hope that this project will be fun, and it will definitely give you hands-on experience building a real NLP system. And I hope that the "competitive spirit" will energize everyone to work hard, be creative, and produce interesting and effective coreference resolvers!