CS-5340/6340 PROJECT DESCRIPTION, Fall 2016

The project for this class will be to design and build your own coreference resolution system. 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. Each coreference resolver will be built by a 2-person 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 accuracy of your system.

INPUT FILE SPECIFICATIONS

The coreference resolution task is to find a *correct antecedent* for each *anaphor*[1] in a document. As input, you will be given the set of anaphora that need to be resolved. Each anaphor will be marked in the original source document with XML tags and given a unique identifier. The format of each XML tag will be: <COREF ID="$">*anaphor*</COREF>, where $ is an alphanumeric string. The entire document will also be surrounded by <TXT> and </TXT> tags, just to make it properly formatted for an XML parser in case you wish to use one.

As an example, consider this news story that appeared on CNN.com:

> One person has been killed in a fire that has burned more than 3,500 acres in the Angeles National Forest and encroached some northern areas of Los Angeles, Mayor Antonio Villaraigosa said Monday afternoon.
>
> The adult male's body was found in a makeshift wood-and-cardboard shelter in northern Los Angeles near the forest, officials said at an afternoon news conference. He appeared to be homeless, officials said; no name or age were released.
>
> Authorities also said they are battling a second nearby blaze, which has already burned 750 acres. Evacuations have been ordered.
>
> The first fire has destroyed several structures, including about 30 mobile homes in the Lopez Canyon area, said Los Angeles County fire inspector Sam Padilla. The mobile homes had been evacuated Sunday.

Figure 1: A Sample Story

Figure 2 shows an XML-tagged version of the same news story. This document contains 10 anaphora that your program would need to resolve. Each anaphor is surrounded by the XML tags <COREF ID="$">anaphor</COREF>, to indicate exactly which words are part of

[1] We will use the term "anaphor" to refer to any noun phrase that has a prior referent in a document.

<TXT>
One person has been killed in a fire has burned more than 3,500 acres in the Angeles National Forest and encroached some northern areas of Los Angeles, Mayor Antonio Villaraigosa said Monday afternoon.

<COREF ID="1">*The adult male's body*</COREF> was found in a makeshift wood-and-cardboard shelter in <COREF ID="2">*northern Los Angeles*</COREF> near <COREF ID="3">*the forest*</COREF>, officials said at an afternoon news conference. <COREF ID="4">*He*</COREF> appeared to be homeless, <COREF ID="5">*officials*</COREF> said; no name or age were released.

<COREF ID="6">*Authorities*</COREF> also said <COREF ID="7">*they*</COREF> are battling a second nearby blaze, which has already burned 750 acres. Evacuations have been ordered.

<COREF ID="8">*The first fire*</COREF> has destroyed several structures, including <COREF ID="9">*about 30 mobile homes*</COREF> in the Lopez Canyon area, said Los Angeles County fire inspector Sam Padilla. <COREF ID="10">*The mobile homes*</COREF> had been evacuated Sunday.
</TXT>

Figure 2: Sample Story with XML-tagged Anaphora

the anaphor, and assigned a unique identifier ($). If a phrase is marked as an anaphor, then it has at least one prior referent in the document. Note that many anaphora will have multiple antecedents. In this case, it does not matter which antecedent your program finds.

OUTPUT (RESPONSE) FILE SPECIFICATIONS

Your program will need to find an antecedent for each anaphor and report this by adding a REF label inside the anaphor's XML tag. The REF label should provide the identifier associated with the hypothesized antecedent. For example, <COREF ID="10" REF="9">anaphor </COREF> means that the NP with ID=9 is the hypothesized antecedent for the NP with ID=10.

Each output file should be named **docid.response**, where **docid** is the numeric identifier of the document. For example, if the input file is **12.crf**, then your output file should be named **12.response**.

IMPORTANT: The first antecedent for each anaphor will not have an ID in the input file that you receive. For example, in Figure 2 the first mention of the victim is "One person". Since that does not have a prior referent, it is not an anaphor so it does not have an ID in the input file. However, the second reference to the victim, *"The adult male's body"*, does have an ID because it is an anaphor (its antecedent is "One person"). Consequently, you may need to assign identifiers to some NPs.

As an example, suppose your coreference resolver determines that *"One person"* is the

antecedent for *"The adult male's body"*. You will need to insert an XML tag around the antecedent NP and give it a new identifier; for example: <COREF ID="A">*One person*</COREF>. This new identifier should also be added as a REF in the anaphor's XML tag (e.g., <COREF ID="1" REF="A">*The adult male's body*</COREF>). An identifier can be any alphanumeric string, but it should not include punctuation marks.

Often, however, your coreference resolver may find that the antecedent for an anaphor already has an identifier (e.g., *"Authorities"* is the antecedent for *"they"*). In this case, you should use the existing identifier for the antecedent when adding a REF tag to the anaphor (e.g., <COREF ID="7" REF="6">they</COREF>). This situation happens when there are more than two references to the same thing. In this case, it does not matter which prior referent you choose. For example, <COREF ID="7" REF="5">they</COREF> would also be correct because "officials", "Authorities", and "they" are all coreferent (remember that coreference is a transitive relation).

<COREF ID="A">*One person*</COREF> has been killed in <COREF ID="B">*a fire*</COREF> that has burned more than 3,500 acres in <COREF ID="C">*the Angeles National Forest*</COREF> and encroached <COREF ID="D">*some northern areas of Los Angeles*</COREF>, Mayor Antonio Villaraigosa said Monday afternoon.

<COREF ID="1" REF="A">*The adult male's body*</COREF> was found in a makeshift wood-and-cardboard shelter in <COREF ID="2" REF="D">*northern Los Angeles*</COREF> near <COREF ID="3" REF="C">*the forest*</COREF>, <COREF ID="E">*officials*</COREF> said at an afternoon news conference. <COREF ID="4" REF="1">*He*</COREF> appeared to be homeless, <COREF ID="5" REF="E">*officials*</COREF> said; no name or age were released.

<COREF ID="6" REF="5">*Authorities*</COREF> also said <COREF ID="7" REF="6">*they*</COREF> are battling a second nearby blaze, which has already burned 750 acres. Evacuations have been ordered.

<COREF ID="8" REF="B">*The first fire*</COREF> has destroyed <COREF ID="F">*several structures*</COREF>, including <COREF ID="9" REF="F">*about 30 mobile homes*</COREF> in the Lopez Canyon area, said Los Angeles County fire inspector Sam Padilla. <COREF ID="10" REF="9">*The mobile homes*</COREF> had been evacuated Sunday.

Figure 3: Sample Output

DATA

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

1. **Raw text files:** the original news articles. These files will end with a **.txt** extension. *Your system does not need to use these files*, but we are making them available because you may find them easier to read.

2. **Input files:** the news articles with XML tags identifying the anaphora that need to be resolved. **These files will be the input files for your coreference resolver and will end with a .crf extension.**

3. **Official keys:** the gold standard answer keys for each news article (i.e., the correct answers that your program's output will be scored against). These files will end with a **.key** extension. These files are for the scoring program to use to compute the accuracy of your coreference resolver's output.

4. **Unofficial keys:** "unofficial" answer keys for each news article. These files will end with a **.fkey** extension. The official keys are difficult to read because they are formatted in a special way for the scoring program. So we are also providing unofficial keys that contain (mostly) the same information but in an easy-to-read format. *You do not need to use these files!* But you may find them useful to more easily see what the correct resolutions should be.

5. **Scoring Program:** we will give you 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 or test new components.

All of these files will be available on our class web page.

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) Initial System Evaluation: For the initial 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. However, your team's coreference resolver **must** produce correctly formatted output and make a good faith (non-trivial) attempt to resolve some anaphora or you will get a zero! As long as your system produces correctly formatted output and *some* good-faith answers (even if just a few answers, and regardless of whether they are correct), you will get at least a 75% (C) grade. 30% of your overall project grade will be based on the performance of your coreference resolver on Test Set #1.

4) Final Evaluation: For the final evaluation, we will run each system on two new test sets: **Test Set #2** and **Test Set #3**. 30% of your overall project grade will be based on your system's performance on Test Set #2 and 30% of the grade will be based on its performance on Test Set #3.

Test Set #2 will consist of documents very similar to Test Set #1. Test Set #3 will consist of documents from a different domain. A system that uses general techniques should work just as well on Test Set #2 as Test Set #3. But a system that has been extensively tailored for documents like those in Test Set #1 may perform better on Test Set #2 than Test Set #3.

I/O SPECIFICATIONS

Your coreference resolver should accept two arguments as input:

1. A **listfile** that specifies the stories to be processed, one file per line. A listfile might look like this:

   /home/cs5340/project/coref/newdata/1.crf
   /home/cs5340/project/coref/newdata/52.crf
   /home/cs5340/project/coref/newdata/33.crf

2. A directory name (string), which is where your program's output (response files) should be placed.

We should be able to run your program like this:

coreference <listfile> <responsedir>

For example, we might run your program this way:

coreference test1.listfile /home/cs5340/project/coref/team5/

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

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 MAY use external NLP software that performs basic NLP functionality, such as tokenizers, sentence splitters, part-of-speech taggers, general-purpose syntactic parsers (as long as they do not include any coreference resolution), and general-purpose semantic dictionaries such as WordNet.

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

EVALUATION

The performance of your coreference resolver will be based on *Accuracy*, which is defined as the percentage of anaphora that were correctly resolved. So if an input file contains 20 anaphora and your system finds correct antecedents for 16 of them, then your system's accuracy will be 80% (16/20).

SCHEDULE

The schedule for the project is:

Friday, Oct. 7 by 11:00pm: fill out the Team Request Form on our project web page (http://www.eng.utah.edu/ cs5340/project.html) and submit it via electronic handin using either the Web handin facility (https://webhandin.eng.utah.edu/) or via the command line: `handin cs5340 teams YourName-team-form.txt`

Monday, Nov. 7: Initial System Evaluation on Test Set #1. The documents and answer keys for Test Set #1 will be released after the scoring is completed.

Tuesday, Nov. 29 (by 3pm!): Final System Evaluation on Test Set #2 and Test Set #3.

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

Wednesday, Dec. 7: In-class project poster session.

Friday, Dec. 9: Project slides/poster submissions 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 Initial Evaluation.
- 30% of the grade will be based on the performance of your coreference resolver on Test Set #2 during the Final Evaluation.
- 30% of the grade will be based on the performance of your coreference resolver on Test Set #3 during the Final Evaluation.
- 10% of the grade will be based on your oral or poster project presentation and submitted slides/poster.

To compute each of these grades, all of the teams will be ranked based on their performance. Clusters will be created for teams that had 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 team's 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!

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. As a point of reference, I am expecting the best teams to post scores in the 40-60% accuracy range, though this is just an educated guess. 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!