Journal Report 8
10/28/19-11/11/19
Bryan Lu
Computer Systems Research Lab
Period 2, White

Daily Log

Monday, October 28

Began sketching out what different cases of objects should return for the dependency tree distance method, and sketched out implementations.

Tuesday, October 29

I coded the method, neglecting the sign of the output (will be taken to be positive whenever possible).

Thursday, October 31

Presentations.

Wednesday, November 7

I finished the dependency tree distance method and began researching the `scikit` library and the logistic learning method.

Thursday, November 8

I realized I had a shortage of triangular geometry test cases (the easiest possible cases to train on), and I started assembling test cases.

Timeline

Date	**Goal**	**Met**
10/21	Complete the extraction of all the necessary features of the sentences needed as input data.	Almost – I have found all of the necessary features except for "dependency tree distance."
10/28	Begin writing code to create a log-linear classifier using `scikit`, and finalize the inputs needed for the algorithm.	Yes, I have all of the features I need per sentence.
11/4	Create the log-linear classifier/learning algorithm and the training data, and begin testing.	I've started, but I've realized this is a rather ambitious task because I still need to finish creating my training data.
11/11	Complete a set of about 40 problems to serve as my training data set, with the correct relations.	N/A
11/18	Begin testing, tweaking previous steps as needed and adding more cases if needed.	N/A

Reflection

The past two weeks have been fairly interrupted, with presentations and the end of the quarter, but I tried to keep progressing as much as I could. I may have fallen a week or two behind schedule at this point, but I hope I can make up the time later.

I faced a decent amount of uncertainty when coding the dependency tree distance method. In the end, I decided to enact the following scheme that seemed to be fairly reasonable, restricting the distances to reals between 0 and 3:

a) If both of the objects passed are explicit, i.e. ABC (a triangle) or AB (a segment), then report the difference in length between the strings, scaled arbitrarily by 0.1.

b) If one of the objects is explicit and the other is an abstract concept, i.e. ABC vs. "circle,", I gave this a distance of 3 by default.

c) If both are abstract concepts, check if both represent "closed" shapes, i.e. circles, triangles, polygons.

   i) If only one does, return a distance of 2.

   ii) Else, check how many relations/objects that are both associated with them. Return 1.5 minus the amount of overlapping relations/objects times 0.1.

I'm not sure how well this will work, but I can always adjust it later.

I've also realized I need more test cases for training, especially with regards to triangular geometry problems like the following, for which it's really easy to create the correct relations for:

Let Gamma be the circumcircle of acute triangle ABC. Points D and E are on segments AB and AC respectively such that AD = AE. The perpendicular bisectors of BD and CE intersect minor arcs AB and AC of Gamma at points F and G respectively. Prove that lines DE and FG are either parallel or they are the same line.

I think I will need to select a decent number of these to use for training my model, which I will do in the next week.