Journal Report 7

10/14/19-10/28/19
Bryan Lu
Computer Systems Research Lab
Period 2, White

Daily Log

Tuesday, October 15

I completed `lexicon.txt`, which now holds all the words I'm looking for in every problem along with the objects and relations associated with them.

Thursday, October 17

I reconfigured detection code to accomodate double-word objects, and to detect any word in `lexicon.txt`.

Monday, October 21

I used the `nltk` library to get part-of-speech tags for every word, which will be used later.

Tuesday, October 22

I calculated an index of every detected word or phrase in the sentence (treated as a list), which will give us sentence distancse easily when needed later.

Thursday, October 24

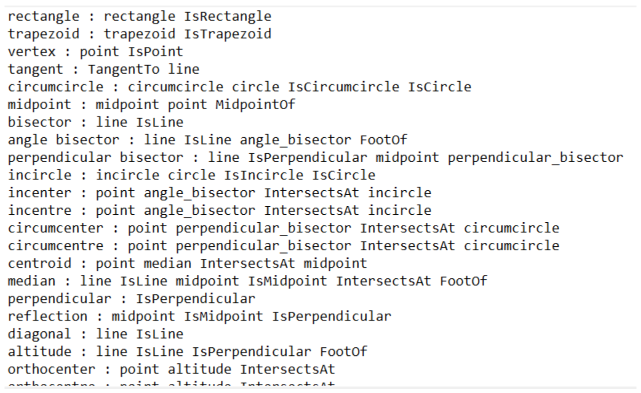
I created `finalrelations.txt`, which identifies the type of inputs, return type, and number of arguments each relation needs in each identified relation in the sentence.

Timeline

Date	Goal	Met
10/7	Create a logical language to give more structure and properties to the objects and relations detected in the problem.	Not quite the goal I needed to meet, but I figured out the path I need to take going forward and started creating the dictionary I needed to have for my problem.
10/14	Finalize the object identification associated with `lexicon.txt`, and figure out how to extract the necessary data to pass to the log-linear classifier.	I completed this successfully, but without much time this week I only started to come up with some ideas as to how to extract the necessary data.
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.	N/A
11/4	Create the log-linear classifier/learning algorithm and the training data, and begin testing.	N/A

Reflection

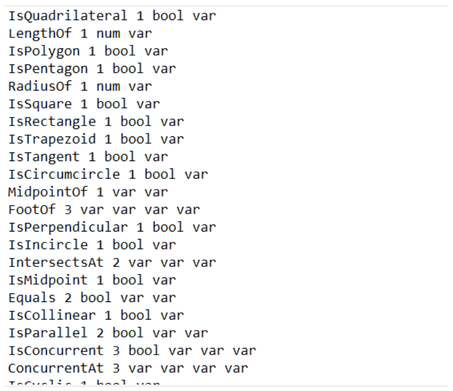
The past two weeks have seen me finish almost all of the necessary setup needed for the first major stage of my project. I can. Here is a snippet of the completed `relations.txt` (for now):



```
rectangle : rectangle IsRectangle
trapezoid : trapezoid IsTrapezoid
vertex : point IsPoint
tangent : TangentTo line
circumcircle : circumcircle circle IsCircumcircle IsCircle
midpoint : midpoint point MidpointOf
bisector : line IsLine
angle bisector : line IsLine angle_bisector FootOf
perpendicular bisector : line IsPerpendicular midpoint perpendicular_bisector
incircle : incircle circle IsIncircle IsCircle
incenter : point angle_bisector IntersectsAt incircle
incentre : point angle_bisector IntersectsAt incircle
circumcenter : point perpendicular_bisector IntersectsAt circumcircle
circumcentre : point perpendicular_bisector IntersectsAt circumcircle
centroid : point median IntersectsAt midpoint
median : line IsLine midpoint IsMidpoint IntersectsAt FootOf
perpendicular : IsPerpendicular
reflection : midpoint IsMidpoint IsPerpendicular
diagonal : line IsLine
altitude : line IsLine IsPerpendicular FootOf
orthocenter : point altitude IntersectsAt
```

Note the two-word phrases, which I had to adapt my code to handle, and the mixture of objects and relations with many of the words. It's easy to add to the list of objects and relations associated with each word I want to detect, and it's fairly straightforward to add more words/symbols I want to detect in the sentence to the text file. I think my code can handle it as long as the word/phrase is at most two words. This versatility is something I will probably want later for ease of modification.

One challenge I will likely face in the coming weeks is the fact that I have to train two (and possibly more!) different log-linear classifiers to identify whether or not a given relation with inputs is valid. Granted, given the properties of the relations, which I have articulated in a separate file (`finalrelations.txt`), most of the relations are one-argument functions:



```
IsQuadrilateral 1 bool var
LengthOf 1 num var
IsPolygon 1 bool var
IsPentagon 1 bool var
RadiusOf 1 num var
IsSquare 1 bool var
IsRectangle 1 bool var
IsTrapezoid 1 bool var
IsTangent 1 bool var
IsCircumcircle 1 bool var
MidpointOf 1 var var
FootOf 3 var var var var
IsPerpendicular 1 bool var
IsIncircle 1 bool var
IntersectsAt 2 var var var
IsMidpoint 1 bool var
Equals 2 bool var var
IsCollinear 1 bool var
IsParallel 2 bool var var
IsConcurrent 3 bool var var var
ConcurrentAt 3 var var var var
```

This means it's plausible for me to handle these separately, but I'm not sure how reliable that will be.

Regardless, I will soon be on my way to create the main machine-learning part of my model, and here is where the test cases and problems I have gathered in the past will come in handy. I have found a website where someone has gathered a lot of problems in one place, so I might consider taking a bit of time on a weekend to scrape it to add to my data set. Currently, I am techinically still on schedule, as next week should be week 2 of 4 of writing and training the log-linear classifier, so I'm optimistic about finishing this important phase of the project on time.