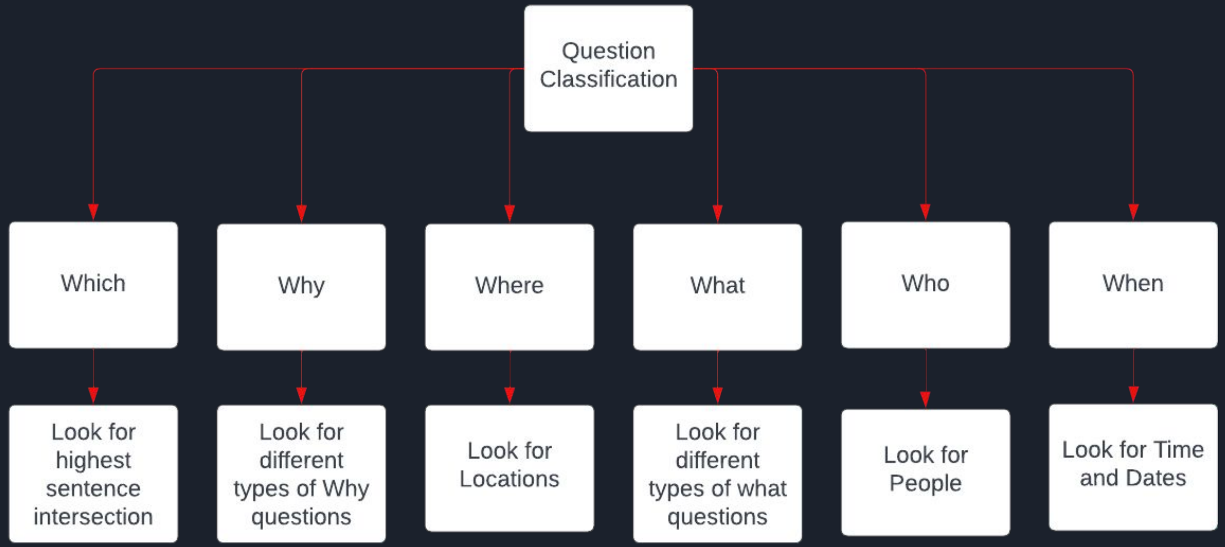
# CS 5340 Project Presentation

Team: The Detectives

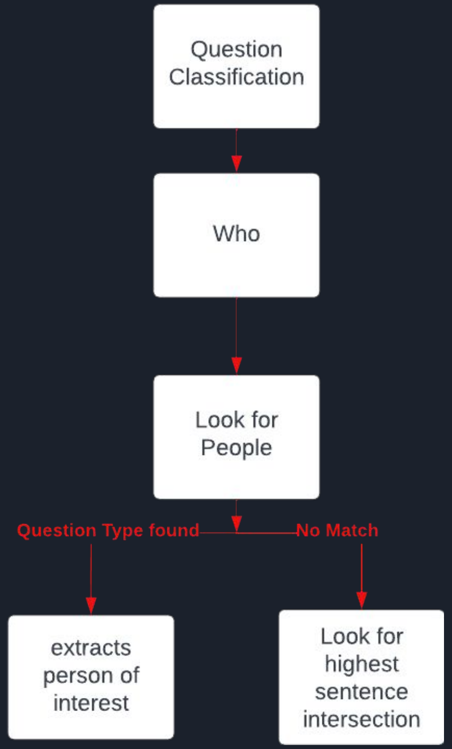
# System Architecture and Algorithms

- Initially we classify questions as who, what, when, why, where and which
- Once the type of question is detected we then loop through the sentences
- With the loop we are looking for candidates that contain pertinent date for the question type


Question Classification
Which
Why
Where
What
Who
When
Look for highest sentence intersection
Look for different types of Why questions
Look for Locations
Look for different types of what questions
Look for People
Look for Time and Dates

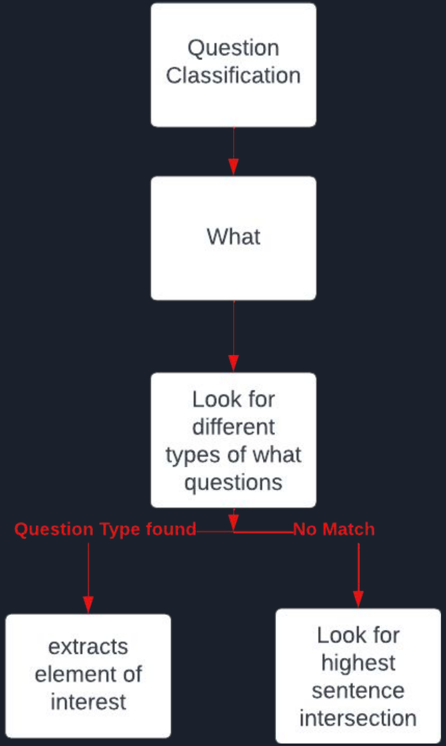
## System Architecture and Algorithms
## Who Questions

- With who questions we look for sentences that contain people
- Once we have a list of sentences we try to narrow down the list by finding common points of interest between the question and the candidate sentences.
- Once the narrowed down to a particular sentence the person of interest is extracted and returned as the answer
- In the event of a tie or multiple sentences the sentence with the highest intersection is found of the candidates and the person of interest is returned
- If no person entities are in the doc then the sentence with the best intersection is returned



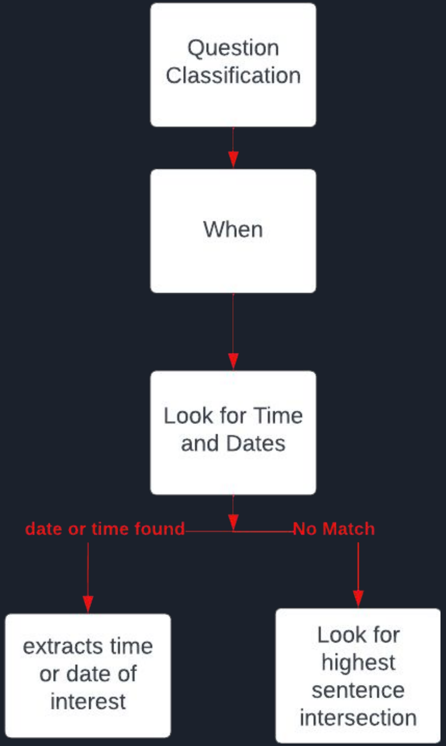
## System Architecture and Algorithms
## What Questions

- With what questions we look for sentences that had common dependencies with the questions
- Once we have a list of sentences we used the dependencies of the question to narrow down the sentences further
- Once the narrowed down to a particular sentence the element of interest is extracted and returned as the answer
- In the event of a tie or multiple sentences the sentence with the highest intersection is found of the candidates and the element of interest is returned
- If no pattern is matched in the doc then the sentence with the best intersection is returned


Question Classification
What
Look for different types of what questions
Question Type found
No Match
extracts element of interest
Look for highest sentence intersection

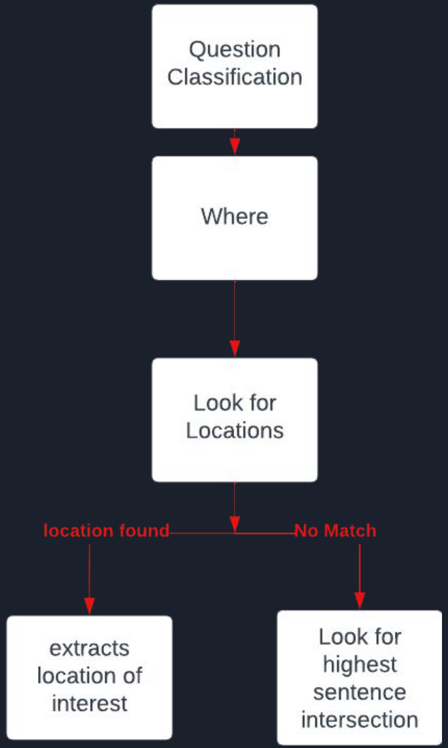
## System Architecture and Algorithms When Questions

- With when questions we look for sentences that contain dates and/or times
- Once we have a list of sentences we try to narrow down the list by finding common points of interest between the question and the candidate sentences.
- Once the narrowed down to a particular sentence the date or time of interest is extracted and returned as the answer
- In the event of a tie or multiple sentences the sentence with the highest intersection is found of the candidates and the date or time of interest is returned
- If no date or time entities are in the doc then the sentence with the best intersection is returned



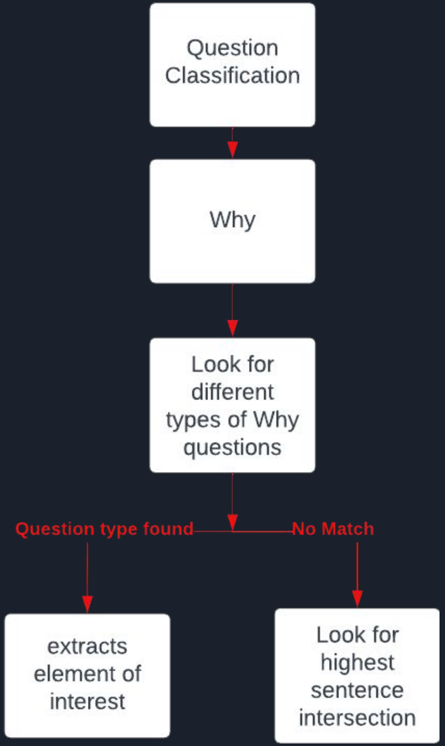
## System Architecture and Algorithms Where Questions

- With where questions we look for sentences that contain locations
- Once we have a list of sentences we try to narrow down the list by finding common points of interest between the question and the candidate sentences.
- Once the narrowed down to a particular sentence the location of interest is extracted and returned as the answer
- In the event of a tie or multiple sentences the sentence with the highest intersection is found of the candidates and the location of interest is returned
- If no location entities are in the doc then the sentence with the best intersection is returned



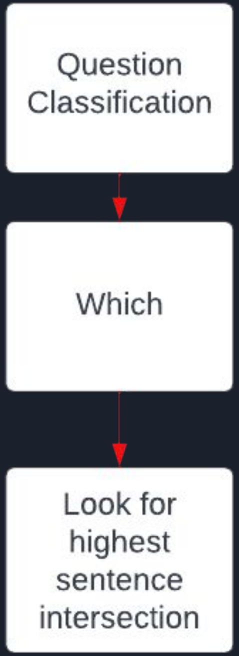
## System Architecture and Algorithms Why Questions

- With why questions we look for sentences that had common dependencies with the questions
- Once we have a list of sentences we used the dependencies of the question to narrow down the sentences further
    - Common terms between questions and answer were used to narrow down the sentences
- Once the narrowed down to a particular sentence the element of interest is extracted and returned as the answer
- In the event of a tie or multiple sentences the sentence with the highest intersection is found of the candidates and the element of interest is returned
- If no pattern is matched in the doc then the sentence with the best intersection is returned


Question Classification
Why
Look for different types of Why questions
Question type found
No Match
extracts element of interest
Look for highest sentence intersection

## System Architecture and Algorithms Which Questions

- With which questions we simply looked for a sentence with the highest intersection and returned the sentence



# Team Member Contributions

- We worked in a paired programing format taking turns as Driver and Navigator during the initial coding
- Then would talk and would each try different ideas to see if we could get a better result the one with the idea would drive while the other watched
- When debugging we went back to the Driver and Navigator roles to iron out errors and edge cases

## External Sources

- Main sources we using concepts from lectures and the class text.
    - https://web.stanford.edu/~jurafsky/slp3/
    - https://my.eng.utah.edu/~cs5340/
- We also used the following python libraries sys, re, math, numpy
- We used spacy heavily to do entity and part of speech tagging
    - https://spacy.io/usage/linguistic-features
    - https://spacy.io/usage/rule-based-matching

## Emphasis/Originality

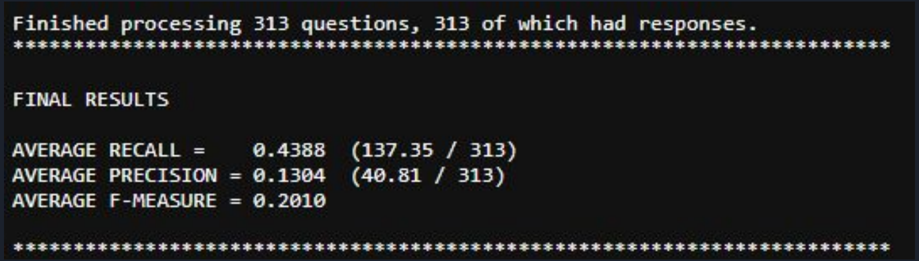
Most of our effort was looking at questions and and the sentences that contained their answer to identify commonalities that could be found between them. These efforts were used as a frame for our rules in the system.

One sub problem we encountered was in narrowing down candidate sentences was difficult when long name was used in the question and occurred heavily in the text we did try to offset this fact using the TF-IDF algorithm from class lecture. Using this and spacy's similarity algorithm we were able to get a better result but it still had some trouble with leaning on very long names.

An aspect of our system that we thought was most creative was our use of noun chunks to try and narrow down the answer to some questions. Also how we used labels of each word through spacy along with some parts of speech tags to find some answers, in our opinion, was fairly creative.

# Performance

Midpoint evaluation results



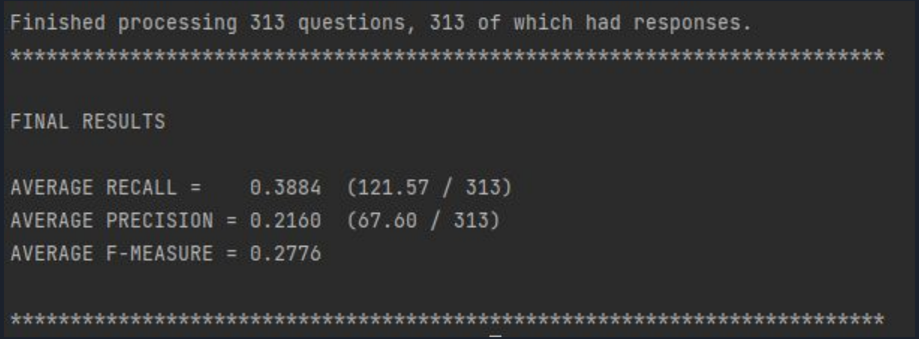
```
Finished processing 313 questions, 313 of which had responses.
*************************************************************************

FINAL RESULTS

AVERAGE RECALL =    0.4388  (137.35 / 313)
AVERAGE PRECISION = 0.1304  (40.81 / 313)
AVERAGE F-MEASURE = 0.2010

*************************************************************************
```

Midpoint results after making improvements



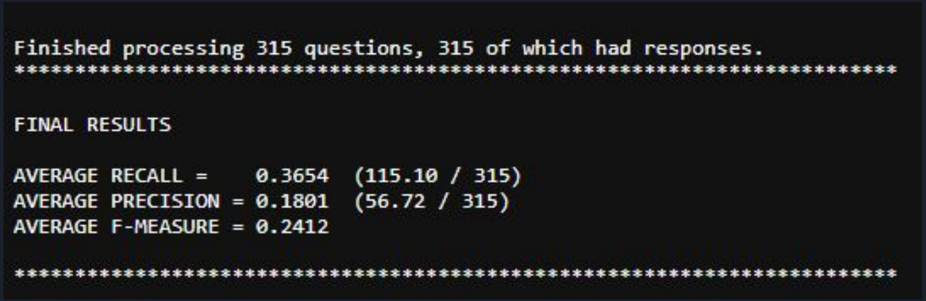
```
Finished processing 313 questions, 313 of which had responses.
*************************************************************************

FINAL RESULTS

AVERAGE RECALL =    0.3884  (121.57 / 313)
AVERAGE PRECISION = 0.2160  (67.60 / 313)
AVERAGE F-MEASURE = 0.2776

*************************************************************************
```

Final evaluation results



```
Finished processing 315 questions, 315 of which had responses.
*************************************************************************

FINAL RESULTS

AVERAGE RECALL =    0.3654  (115.10 / 315)
AVERAGE PRECISION = 0.1801  (56.72 / 315)
AVERAGE F-MEASURE = 0.2412

*************************************************************************
```

Our system worked better with "who" questions because it was the one type of question that was simplest and one that we worked on the most to get us started. Our system also performed better with recall because in most cases we thought it best to return the entire sentence.

## Lessons Learned and Successes/Regrets

One regret from this was not spending more time reading Spacy documentation. Spacy has a lot of different functionalities that would of helped to with implementing the rule based system. These functionalities we found rather late in the project and was not practical to rework program to leverage them.

One thing that helped was switching from Spacy's `"en_core_web_sm"`to `"en_core_web_lg"`this greatly helped better tag entities and allowed us to get better data.

One thing we came to sort of regret was not having more time to work on our system as we focused so much on returning the right sentence that we ran out time to work on narrowing down the answer within the sentence for some questions.

Something we learned was to give equal consideration for all aspects of a project instead of hyperfocusing on one aspect of it and realizing that we had run out of time to effectively do the other parts.