CS 4740/5740 – Project 2 Proposal
Team members: Darpan Kalra(dk557), Nithish Raja Chidambaram(nr293), Tommy Shum(ts539)

1 - Baseline System

1a - Baseline Implementation Details

Our first baseline system uses naive string matching to find the sentence in the paragraph(context) that matches the most words in the question string. In the testing file, we will preprocess the data by removing all the punctuation, whitespace, articles, and lowercase the text. Since we are using naïve string matching, we only care about the word tokens.
Our system, splits the paragraph into chunks of 5 words. We compare each of the 5 chunks in the paragraph to the question, to see how many matching words there are. We process the entire context in this way, moving one word ahead in a sliding window fashion, and considering 5 words at a time.
To do this efficiently, we have a dictionary of words in the question string, and count the number of match words in the context. We return the chunk with the maximum matching count as the answer. If there is a tie in the counts, we just choose the first chunk string as the answer. Of course there is a lot of room for improvement over this method.

1b - Baseline Motivation

Since the answer is supposed to be found in the text, a naïve string matching approach seems like a good starting baseline system. Matching words can help us locate the relevant part of paragraph. We looked over the answers in the training file and saw that on average the answer length is five words, so we chose a sliding window of size five. Later, we also experimented with various values for the size of the window as well.

1c - Baseline Results

Our baseline system is performing poorly because we are performing naïve string matching which tries to match words in the question to words in the paragraph. The answer in the paragraph might be to the left or right of matching phrase. For example, if we have the question "When was Notre Dame University founded ?" and there is a sentence in the paragraph "Notre Dame University is located in Indiana, it was founded in 1899". The answer is all the way to the right of sentence, which won't be returned as the answer. Instead the most amount of overlap between the sentence and the question is "Notre Dame University is located" and this will be returned as the predicted phrase. We indeed see a lot of such poor predictions over the test set, because of which the score of our system is poor. The results of our systems for different sizes of the window are as follows:
Although we present results on both development and testing sets, for our naive system it doesn't matter as we don't do any training or learning.

<u>On development set:</u>

Window Size	Exact Match Score	F-1 score
10	0.0189	14.5280
7	0.0473	12.2595
5	0.0756	11.7533
3	0.1419	6.3317

On testing set:

Window Size	Exact Match Score	F-1 score
10	0.0245	11.4912
7	0.0081	9.4589
5	0.0326	6.9448
3	0.0570	4.1212

We see an interesting pattern, that for larger window size we see better F-1 score, and for smaller window size we see better exact match score. It kind of makes sense as well, as F-1 score takes the average match percentage so the larger window size, greater is the potential for match. On the other hand, for exact match score, smaller window size is performing better because on average the answers are indeed of short length.

2. Proposal

We will try to use Intent Classification to extract the intent of the question. We thought of using advanced techniques like a multi-class machine learning classifier to classify the intents, but ran into the problem of not having labeled training samples for that. So instead a rule based intent classification by looking at the question's start phrase to identify the type of wh-question (What, who, why, when, how, etc) is proposed.

Next, we will try to use Named Entity Recognition to recognize some of the named entities in the paragraph(context). Now that we have the intent of the question and NER tagged words, we return the phrase with NERs that correspond to the intent of the question. For example, if the intent of the question is identified to be a place, then we look for B-LOC NER and match the string till the last I-LOC NER.

3. Workflow

- Tommy wrote code to parse the JSON files and preprocess the data
- Nithish and Darpan wrote the code for the string matching baseline system together
- All three members worked on the proposal together