Summary of progress so far:

What we have done

We have made significant headway on our project. We’ve manually created a dataset of 64 relevant critical reading questions that deal with word disambiguation. We’ve created a pipeline to automate the reading and storing of the data and information, with the option to evaluate on either dev or test datasets. We’ve also automated loading of glove vectors through the command line.

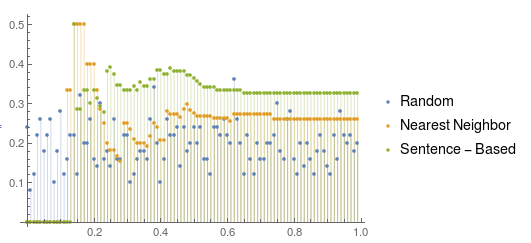
We’ve created a scoring mechanism and have tested our baselines. Our baselines are as follows:

|  |  |
| --- | --- |
| **Baseline Model** | **Score** |
| Random | 220 |
| Nearest Neighbor (NN) | 300 |
| Average Context Sentence (ACS) | 360 |

In the Random Model, we chose a random answer. In the Nearest Neighbor model, we chose the answer with a glove vector (distance metric: cosine) most similar to the target ambiguous word. In the Average Context Sentence Model, we averaged the glove vectors of all the words in the sentence and chose the nearest neighbor (distance metric: cosine) among the answers.

When we parameterized the models and abstained from guessing under a certain confidence threshold (i.e. answer vector to target word vector distances were too great), our ACS Model scored a 380 with a cosine distance function and a .45 threshold.

We also analyzed whether this ACS Model was on the right track. We graphed the precision of the ACS on different thresholds (recall) and produced the following graph:



This graph details the parameter threshold distance (cosine) on the x-axis and the percentage of correct guesses on the y-axis. Underestandably, the Random model performed randomly, centered around .2 (1 out of 5 chance). The NN Model did well with high levels of confidence, but only relatively so, as it never got more than 50% of the answers correct. The ACS Model’s (Sentence-Based in the Legend) precision did not increase with higher levels of confidence. This suggests that averaging the glove vectors of the sentence is not a correct way to approach this problem.

What we still need to do

Moving forward, we need to examine better models for disambiguating words. Here are our proposed models:

1. We may try to use the tone of the entire passage to influence our choice of answers
2. Using Tf-Idf to choose the important word vectors in the sentence to average (or add)
3. Using Tf-Idf scores to choose words
4. Using PPMI scores to choose words
5. Integrating the disambiguate function from distributedwordreps into our code and attempting to identify which disambiguation is the correct one
6. Guessing the word or vector that might be used instead of the target word and using that vector to identify the right answer (official advice for this type of question when taking the SAT).
7. A combination of any of these models.

Any obstacles or concerns

I have two main concerns moving forward. The first is that we have so many models to build, to tune and to combine that in all likelihood, we will not come upon the most optimal solution to this problem. While we have designed a seamless pipeline, data format and scoring measure, we still have to implement these models, which is the bulk of this process. The second concern I have is our SAT Question dataset. We have broken down and extracted questions from numerous SAT Critical Reading sections, but we only have a dataset of 64 distinct questions. We are aiming for 100, but attaining the sample size is an arduous and time-consuming goal.