Peter Valentine
CIS-700 Deep Learning and Theorem Proving
3/8/2022
Homework 1 Report

NOTE: I wanted to add a quick disclaimer that I worked with Yehya Farhat on this assignment. We discussed ideas for the ranking heuristic with one another, just bouncing ideas off one another. The implementation of my algorithm however was done completely on my own.

To start out I decided to try what I called long rank, that is the opposite of short rank which was our baseline heuristic. I got the longest instead of the shortest, and as expected it did not go well. For every test case, both the size of 2 and 3, the number of steps and the run time of the algorithm were worse in all cases. Although I expected this to be the case, I decided to try it just to see what would happen, and to verify that short rank was in fact a good algorithm for premise selection.

My second idea was more sophisticated than long rank, so I decided to choose pairs that were not similar to one another. The idea was that the algorithm would compare the atoms in p with the atoms in q, and for every atom x in p, if that atom was also in q then the “rank” of that pair would be incremented by one. That would be run for every pair p and q, and at the end, the lowest rank would be chosen. This would mean that the pairs that were the least similar to one another would get the lowest rank. This however also proved to not work quite as well as I had hoped. After thinking about this for a while some issues with this approach became clear to me. Let's say for example, you have two atoms, p and q, and you have the following pairs:

(p) (q) and (p) (~p)

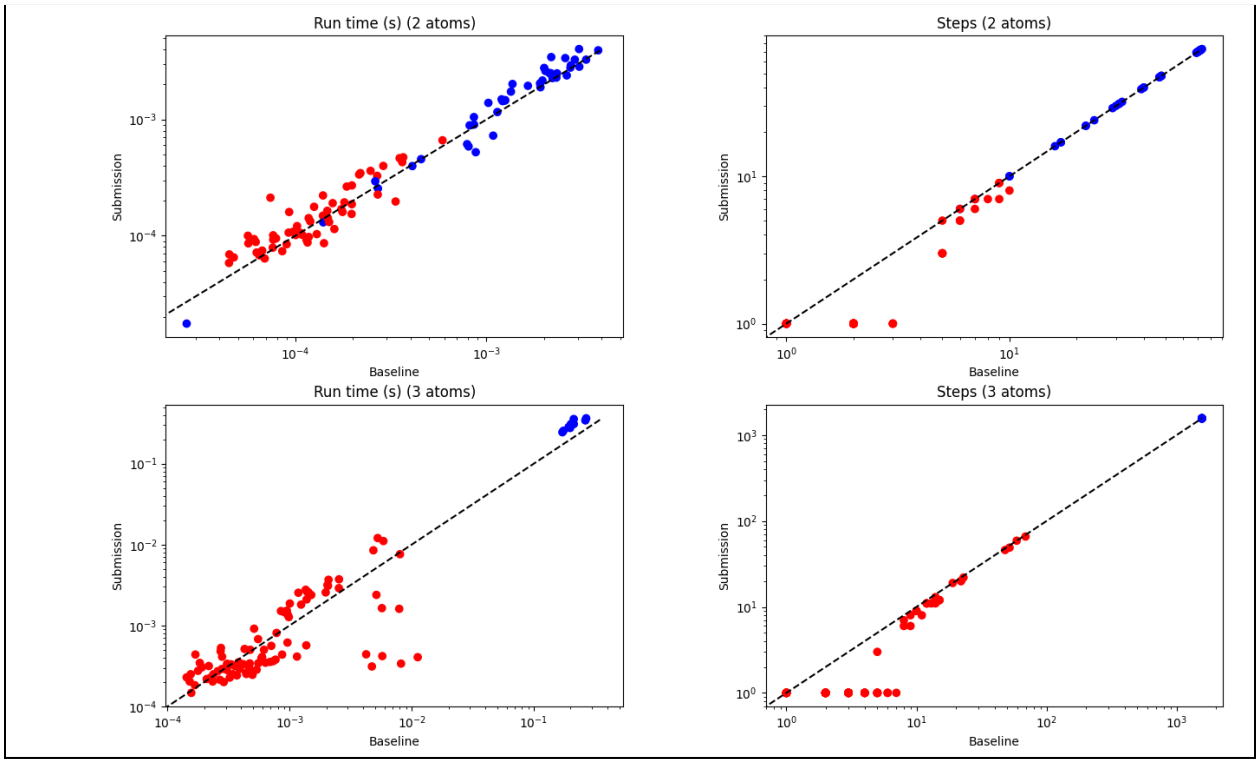
Where the ~ is not, it would actually rank those two pairs as being the same in terms of dissimilarity. However one pair will quickly get you to a contradiction and the other will not. This also brought into light another issue, that is that the algorithm does not take length into account at all. It might be more beneficial to look at pairs that are shorter even though there are two sets of pairs that are ranked the same, as we see that the short rank heuristic is a good baseline. So at this point, I knew I needed to keep looking at the shorter pairs.

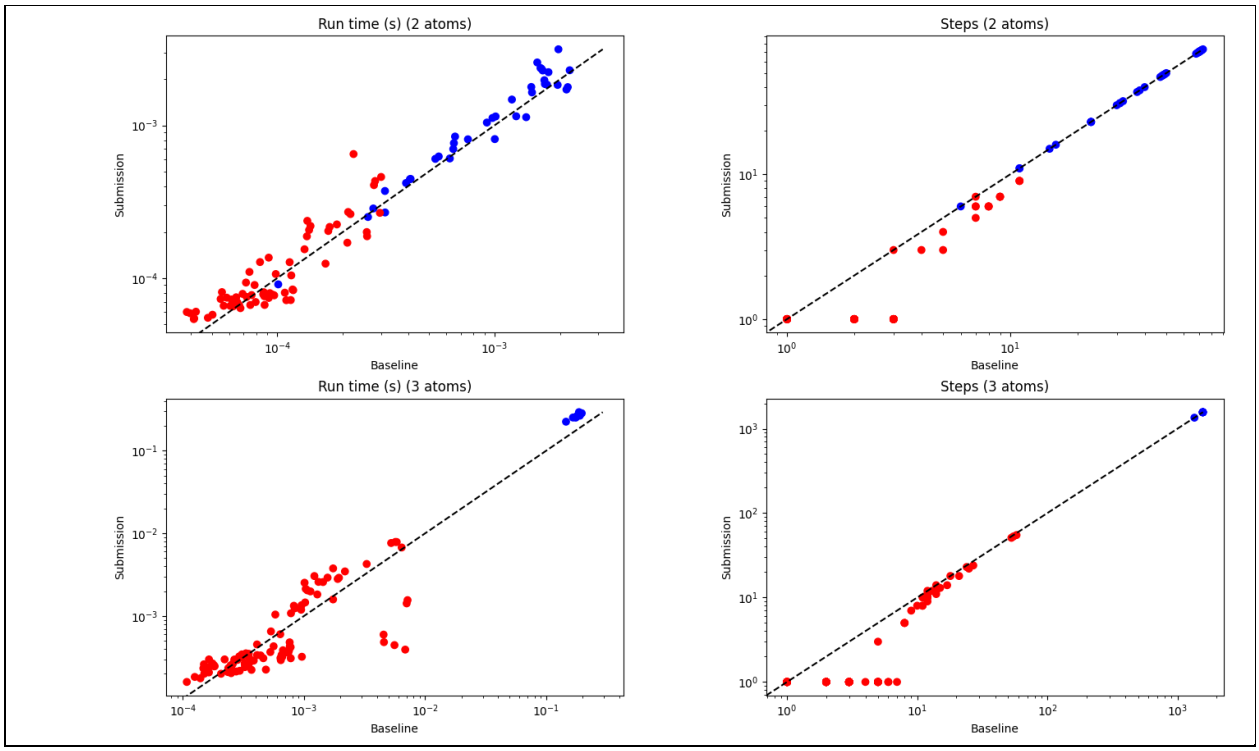
After some discussion it seemed like a good idea to try finding inverses of atoms instead of just dissimilar atoms. The idea is to only add to the rank if you find an atom’s inverse in the other pair. For example in the following pairs:

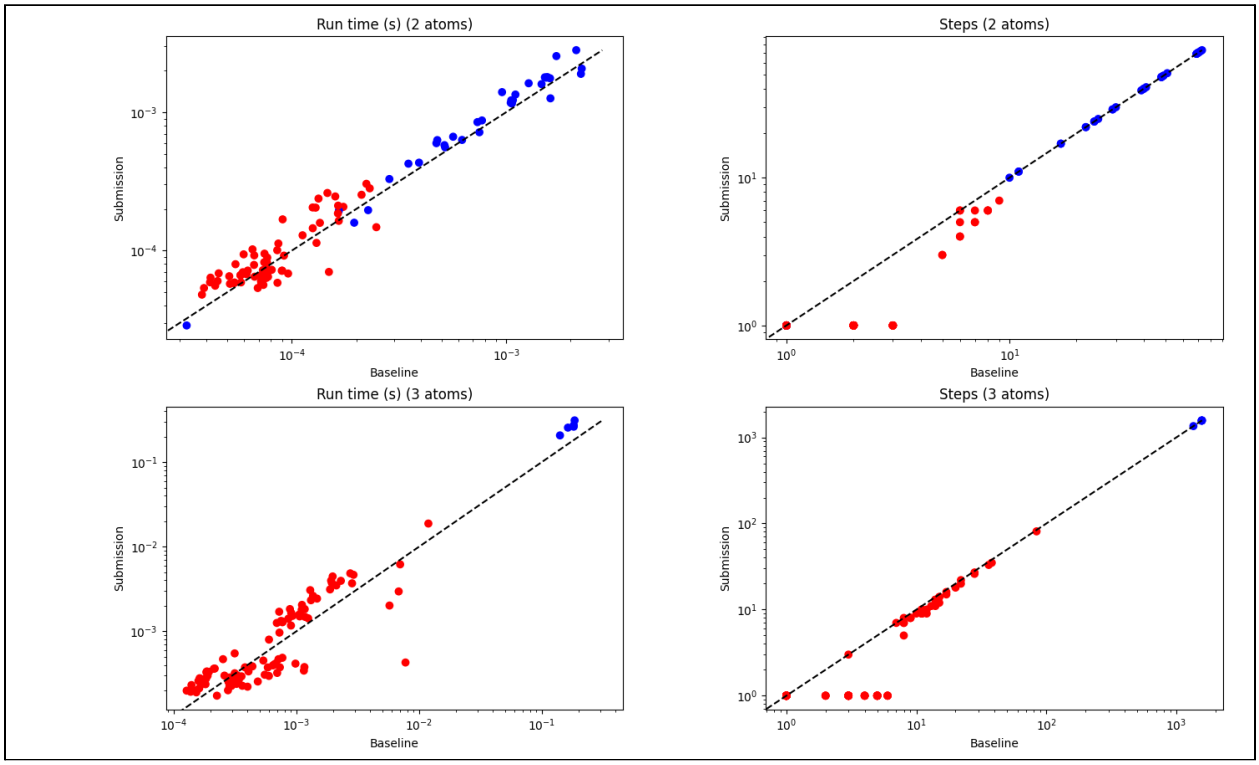
(p) (q) and (p) (~p)

P and ~p would get incremented while p and q would not. This coupled with looking at the shorter pairs gave good results. The results were that I was able to beat or tie the baseline on steps for both two and three atoms, which can be seen in the images below. In addition to this, I

was also able to beat the baseline on time sometimes, which can also be seen in the images below. I was not able to beat the time all of the time however, especially in the case of the statements that did not have contradictions (the blue dots). Overall this heuristic was able to beat the baseline on steps all of the time, and in terms of time more could be done to optimize the code.




Run time (s) (2 atoms)
Steps (2 atoms)
Run time (s) (3 atoms)
Steps (3 atoms)
Submission
Baseline


Run time (s) (2 atoms)
Steps (2 atoms)
Run time (s) (3 atoms)
Steps (3 atoms)
Submission
Baseline