2020 Spring CS372 Homework #1

20190046 권해찬

In this assignment, we were given a task to find pairs of expressions of which D increases the intensity of a restricted group of lexical items E. My approach to this task ranks bigrams of a given text with respective scores that are calculated based on the core characteristics that define the given D and E pairs. In this report I will try to explain why and what makes this approach yield a result of reasonable quality. I’ve selected features of such pairs that should be considered for this task, and have attempted to include bigrams that satisfy these three essential features:

1. D modifies E.

2. D only modifies a restricted group of lexical items E.

3. When D modifies E, the semantic intensity increases.

For the first task, I have tried to utilize part of speech tags to ensure that the first word in the bigram does indeed modify the second word. When a word modifies its next word, the part of speeches of the respective words are usually adjective-noun, adverb-adjective, adverb-adverb and so on. The point is that the part of speech sequences can be inspected in order to verify if the the first word modifies the next one. I have filtered bigrams with relevant part of speech sequences, and double checked the part of speeches with nltk’s BigramTagger to filter out bigrams that include ambiguous part of speech sequences.

For the second task, I construct a ConditionalFreqDist object from the list of filtered bigrams that I’ve retrieved from the first task. The *modifying* words would be the conditions in the object, and the *modified* words would be stored in each conditions. Since we’re trying to retrieve modifiers that only modifies a restricted group of lexical items, I filtered out conditions that have many items in it, since having many items under a single condition signifies that the modifier can modify a wide range of words.

For the third task, I begin with the filtered bigrams from the ConditionalFreqDist in the second task. From here all of the bigrams are possible candidates for D and E pairs. To distinguish the ranks between the candidate bigrams we have, I grade each pairs based on two factors: Does the intensity of the word increase with the modifier? And how restrictive are the words that the modifier modifies? To turn these criteria into a tangible score, I use SentimentIntensityAnalyzer to detect change in intensity, and the size of a given modifier’s FreqDist in the ConditionalFreqDist object. Briefly speaking, the scores are calculated proportional to the amount of difference in intensity and inversely proportional to the size of a given modifier’s FreqDist, in order to take the factor of only modifying a few words into consideration. Then we have our bigrams each with its respective score, and when we sort the bigrams according to the scores, we have our final result.

Since my version of the solution thoroughly covers the three important aspects of pairs D and E, the resulting output can definitely be considered as of reasonable quality. There are, however, improvements that can definitely be made to this solution.

First of all, this solution constructs a ConditionalFreqDist object from a pool of bigrams only from the Brown and MASC tagged corpus since they were the only few among many other corpora that offered a tagged version of all the words in the corpus. Only if there were more samples to work with, a variety of more expressions would be available at hand, which in turn would possibly yield expressions with better scores.

Also, there was a compromise in calculating the semantic intensity because SentimentIntensityAnalyzer was utilized to fulfill such needs. What this solution actually needs is an analyzer that measures change in semantic intensity in general, not necessarily a sentiment intensity analyzer. After long hours of research in an attempt to find tools that analyze semantic intensity, I’ve reached the conclusion that such methods are out of my capabilities for this homework. I had no choice but resort to the closest tool which was vader’s SentimentIntensityAnalyzer. With the right tools to detect change in *semantic intensity*, the solution will yield far more accurate results.

Finally, the results of this method would improve if frequently used bigrams were to be prioritized, which can be said to be an extended problem from the first one. Although not incorrect, there were occurrences of unfamiliar expressions in the results, which made the results seem unfamiliar overall.