CS372 HW2 Report

20180368 An Haechan

**Result discuss& Improvements**

The result was fair enough yet there are still some non-intensity modifying that changes meaning of the word. I tried to exclude those using my algorithm: how the noun get modified by the (ADV+ADJ) pair, is modified by single-word ADJ in the corpus and test the similarity of single-word ADJ and pair’s ADJ to rule out meaning-changed modifiers. This is partly due to the fact my method for finding noun that gets modified by adjective is not perfect, in fact, it only cares frequent cases. Limitation and further improvement of my algorithm(it only counts ADV-ADJ pairs, …etc) will be discussed in the later part of report a lot(they are numbered 1,2,3,4,5).

About the ordering : I think it is well-ordered in terms of uniqueness. I defined uniqueness of a modifier to ‘How small is the set of lexical items that certain modifier modifies?’ So it doesn’t account on ‘how frequent it is used in the corpus’, rather ‘how many types of words it modifies’. In consequences, for the initial 100 results they all only have one word that they modify, having ‘equal uniqueness’. But if you see the entire result, frequent modifiers has number of modified words(although due to my algorithm many frequent modifiers are filtered out already). According to my definition of uniqueness, I think the result is good enough.

My code only considered (ADV+ADJ) pair, because other pairs like (NN+ADJ : rock hard), (ADJ+NN : stark contrast) are more challenging to find. For (NN+ADJ) pair, it is rare that noun is used for modifier, for (ADJ+NN) pair, there are too many cases that ADJ is not used for modifying only intensity. Therefore considering (NN+ADJ) , (ADJ+NN), or even (VB+ADV) pair dulls down correctness. However, if there are enough time and corpus**, 1) I can make my algorithm more fine-grained for finding (NN+ADJ), (ADJ+NN) pair** (Fine-grained is meant by such as path similarity >0.3 to >0.5) so that only pairs that are really ensured to be intensity-modifying pair to be chosen as output. It can compensate the fact (NN+ADJ), (ADJ+NN) pair decreases the correctness.

In current corpus, if some expression is desired one like ‘NN is leather hard’, my code inspects that are there any other expression ‘NN is (ADJ)’, which has similar meaning with ‘hard’. In result, if corpus is small so there’s no expression ‘NN is (ADJ)’, we cannot assure ‘leather hard’ is intensity-modifying or not. So if I use **2) larger corpus and time, there will be another expression with higher possibility.** Moreover, if I have larger amount of time, **3)change checking from ‘similar adjective is used for the same noun’ to ‘similar adjective is used for the similar noun’** using Wordnet.

**4)Improving the algorithm for ‘finding modified noun’** can also be one way. I only considered frequent, limited patterns of modifying such as (ADV+ADJ+CC(and, or,..)+ADJ+NN), (ADV+ADJ+NN). Initially, I wanted to use regular expression I learned in the class. However, we have to deal with the token with tag. It was very frequent to find expressions like ‘NN is good’, mixed with word and tag. So it wasn’t possible to use default nltk token-wise regexp finding function. But if I implement one, it will improve my algorithm a lot.

**5)Also handling proper noun and pronoun.** As my algorithm searches how the noun get modified by the (ADV+ADJ) pair, is modified by single-word ADJ in the corpus (to verify the ADV is only used for intensity-modifying), I need to stick with common nouns. However, if I can implement algorithm that finds similarity between proper nouns and finds the noun that pronoun means, I will be able to consider more expressions too. However, I’m worried about it would be a little too expensive.