CMPT 413/713: Natural Language Processing **(Due: 10/12/21)**

Programming Homework #2

Group Members: Joshua Malmberg, Hewei Cao, Keenan Byun
Group Name: Mochi

1 Objective

The aim of this assignment is to implement a lexical substitution algorithm with semantic retrofitting. Lexical substitution is the task of predicting a replacement for a target word in a sentence, based on the context in which it is used. Given a dataset contain word vectors for every in the vocabulary, a replacement word can be selected by taking the word whose word vector is most similar to the target word's word vector. In this instance, we use cosine similarity to compare two vectors. For this assignment, the word vectors and synonym sets for the vocabulary are provided. We retrofit the word vectors to match the semantic information stored in the lexicon, producing a more effective lexical substitution model.

2 Method

Initially, we implemented a baseline retrofitting algorithm based on the example algorithm provided in the assignment description. For the baseline, we set all the alpha and beta retrofitting parameters to 1 and the time steps to 10. To access the synonym sets from our program, we used a Python dictionary, with each synonym set stored in a key-value pair. The member of each synonym set listed first in the text file was used as the key, and the remaining words composed the value. When retrofitting, our baseline algorithm searched for each word in the keys of the dictionary and updated that words word vector if it was found. This approach does not fully utilize the semantic information in the synonym sets, leading to poor performance. We attempted to address this by creating a second dictionary to record which sets each word belonged to. Using this data structure, the retrofitting algorithm can efficiently access all the relevant synonym sets for each word. The baseline model was also limited by the alpha and beta parameters used when updating each word vectors. Ideally, when updating word vectors there should be a balance between the contributions of the old word vector and the synonym word vectors. However, some words may have an immense number of synonyms, in which case the contribution of the synonyms grossly outweighs that of the old word vector. To address this issue, we enforced the condition:

$$\Sigma_{j:(i,j)\in E}\beta_{ij} = c\alpha_i, \forall i \in V, constant\ c \in R$$

After this, we further tweaked the parameter c, iteration count T, and lexicon(s) used to maximize performance. Additionally, we created a second word vector file to store the new word vectors separately such that they would not be used when updating other word vectors in the same time step. Finally, we altered the similarity metric to use context words in addition to the target word when predicting a replacement.

3 Results

	Dev Score
Baseline	43.98
Full Graph Best Segmentation	37.52
Partial Graph Best	50.03

We found that using a second data structure to access all the information in the synonym graph actually decreased performance. We obtained the highest performance using the partial graph, balanced α and β parameters, parameters $c = 0.5, T = 25$, and only the Wordnet lexicon.

4 Contributions

Hewei Cao: Tested which lexicon and parameters provided best performance, improved retrofitting efficiency, debugged baseline; **Keenan Byun:** Implemented context substitution and debugged baseline, composed Jupyter Notebook detailing results; **Joshua Malmberg:** Implemented baseline retrofitting algorithm and wrote report.