README

bgu: lioryi | github: lior-yishay | 211810650 | ליאור ישי
bgu: shaunshu | github: shaun-git-hub-star | 325114569 | שון שוסטר
bgu: idoman | github:idomandl | 213479926 | עידו מנדל

*Dependency path = not including edges
*Full dependency path = includes edges
*Noun pair = The first word and the last word of the full dependency path.
*the key and the value are separated with a comma ‘,’

The Algorithm Steps:

1. **Filter irrelevant dependencies** - map-reduce

- Map:
    - Input: syntactic 3 gram | records = 1870002
    - Output: <dependency path>, <noun pair> <count> | records = 571319
- Reduce:
    - Input: <dependency path>, list[<noun pair><count>] | records = 571319
    - Output: <noun pair>, <dependency path of at list dp_min unique pairs> <count>
      records = 193985

2. **List the relevant dependencies** - map-reduce

- Map:
    - Input: output 1 - <noun pair>, <dependency path> <count> | records = 193985
    - Output: <dependency path>, <> | records =193985
- Reduce:
    - Input: <dependency path>, list[<>] | records= 193985
    - Output: <dependency path> | records= 867

3. **Create training vectors** - map-reduce

- Setup:
    - Load and index the relevant dependencies from output 2
- Map1:
    - Input: output 1 - <noun pair>, <dependency path> <count>
    - Output: <noun pair>, <”table1”> <dependency path index> <count>
- Map2:
    - Input: hyprenym.txt - <noun pair>, <label> | combined records = 336510
    - Output: <noun pair>, <”table2”> <label> | combined records = 336510
- Reduce:
    - Input: <noun pair>,
      list[<”table1”> <dependency path index> <count>] + [<”table2> <label>]
      records = 336510
    - Output: <noun pair>, <training vector> <label> | records = 788

4. **Calculate classifier** - not done with map-reduce

   We chose to use the WEKA library for this process.
   In order to create the classifier, we first needed to convert the previous output into csv then arff and finally to an "instances" object. After that we have a data object that we can train our classifier on.

   We chose to use the "Random Forest" machine learning algorithm to calculate our classifier. We used 10-fold cross validation on the "Instances" data object, to train the classifier and to calculate the F-Measure, Recall and the presion values.

Classification of new words:
In order to classify whether two words have a hypernym relationship in, We take the respective vector from the "CreateTrainingVectors" data set, convert the vector to an "Instance" object and we run it via the classifier.

Future improvements:
We can save the classifier by serializing the Classifier object, for example using JSON. In addition we can convert the "CreateTrainingVectors" to a database, based on key value. For example "MongoDB", which can improve the runtime for classifying an unclassified hypernym. In addition we can save our results to improve the runtime.

The output with the f-measure, recall, and all the examples for TP, TN, FP, FN can be found inside the html output file.