Erik Zorn

Dr. Buell

Assignment 2

28 February 2019

Dictionary to Thesaurus

The goal of this program is to leverage the GNU dictionary to generate a thesaurus using text processing techniques. The python script operates off of the assumption that if the definitions of two terms share enough words in common, then it is likely that those two terms are synonyms of each other.

The first step of this program was to scrape the GNU dictionary to extract useful information. The GNU dictionary was formatted in a way such that there are twenty-six files, each containing terms starting with a single letter of the alphabet. Words beginning with numbers or punctuation are grouped with the terms beginning with “A.” The files are formatted in an XML style with tags indicating what each word group represents. For this project, only the words and definitions are considered. Words and definitions were located by using simple regex to capture everything between “<ent>” tags as the word and everything between “<def>” tags as the definitions. These word-definition pairs are stored in a python dictionary as key-value pairs.

When considering how to store these word-definition pairs, one must handle words with multiple definitions. The approach I chose for this was to store each definition with the word separately, putting a unique identifier on the end of the word for each definition to avoid duplicate keys. For example, the word “fly” has several definitions and is stored as “fly” : <definition 1> , “fly\_1” : <definition 2>, “fly\_2” : <definition 3>, and so on. This approach was chosen as opposed to storing all definitions under the same key because one word may have two definitions that together do not reflect what the word itself means and therefore would generate unrelated synonyms.

A final step of normalization needs to occur before the thesaurus is ready to be generated. I chose to convert the definitions to lower case and then remove all of the stop words. Stop words include but are not limited to words such as: "i", "me", "my", "myself", "we", and "our." This is done because it forces more significant words in definitions to have to match in order for two words to be considered synonyms. This dictionary is complete and exported to a JSON file so that it does not need to be recreated each time the program is run.

At this point, the logic to create a thesaurus is easier when pulling from the JSON file that was created. A function takes in the dictionary and a threshold representing what percentage of definition words must match. Then nested for loops are used two compare every term-definition pair to every other term-definition pair. The definitions to compare are made into sets and the shorter length is recorded. Then the intersection is found and that length is recorded. If the length of intersection divided by the length of the shorter definition is above the threshold, then the words are considered synonyms. This is appended to a synonym list and the next word is processed. Once a word has been compared to every word in the dictionary, the list of synonyms is added to a dictionary with the word as the key. This continues for all words in the dictionary.

I ran this program with a threshold of 80% word match. Looking at the words “Adolescent” and "Adminicle” the following synonyms were generated:

"Adolescent\_1": [

"Ageless",

"Allowance\_5",

"Child\_3",

"Destructive",

"Diversion\_1",

"Forwardness\_2",

"Greenness\_2",

"Hoiden",

"Hot-brained",

"Implant",

"Inexperience",

"Juvenal",

"Mistaught",

"Mister",

"Pliable\_1",

"Precipitance",

"Pregnant\_2",

"Redemptorist",

"Sinewed",

"Tame\_2",

"Toward\_2",

"Unthinking",

"Verdant\_1",

"Youngth",

"Youthful\_2",

"Youthhood",

"out-of-school"

],

"Adminicle\_1": [

"Preef",

"Preve\_1",

"Prief",

"Probate"

],

The appropriate synonyms are left unformatted, the related words are highlighted yellow, and the irrelevant words are colored red. I categorized these by quick google searches of the definitions and used my best judgment to group them. This is subjective and others may consider things differently. The counts of groups are: 10/27 correct, 6/27 related, and 11/27 irrelevant for “Adolescent”. For “Adminicle” all synonyms appear to be appropriate. This is most likely because the definition for “Adolescent” after stemming and removing stop words is just the word “Youth,” which will be considered a synonym to any other word whose definition contains the word “Youth.” The definition for “Adminicle” is "corroborative explanatory proof." Because this definition is longer, it is likely to yield more appropriate results. It must also be considered that if any of the synonyms have short definitions that could play a role in why a match was made.

There are definitely some errors in this program that can be explained. Two errors in particular are the vast number of words with no synonyms and the number of words with inaccurate synonyms. When glancing over the generated dictionary and cross referencing it with the thesaurus, it is sometimes apparent why errors occurred. In many cases, definitions will contain only 1 word because the rest were removed as stopwords. This causes definitions to meet the threshold if only that 1 word matches. If the one word in the definition is a common word, then it may match a lot of other words and if it is uncommon then it will not match any.

This assignment could be tested with other techniques to check for better results. One technique could be using the transitive property to a certain degree to say that if two words have a synonym in common, then they must be synonyms themselves. Results could also be checked with an adjusted stopword list to cut out less words. This may allow the threshold to be more precise because definitions will be longer. These tweaks were hard to test because of the run time of this program. An attempt at multi processing was made but did not significantly improve run times. With more testing, hopefully this program could yield more accurate and consistent results.