|  |
| --- |
| Stevens Institute of TEchnology, Hoboken, NJ. |
| Verso Dictionary |
| A Reverse Dictionary |
| By, |
| **Kishan J** |
| **5/11/2016** |

|  |
| --- |
|  |

Abstract:

Verso dictionary is developed using Python. This script will help the user to get the meaning to his definition. When the user is writing an article and if he is unware of the particular word he want to use, this script will help him guide to the best possible word to finish his document or article

This is a prototype version. There are few limitations in this particular script. This script is developed to explain how we can work around with the words and give the user what he wants.

This product can be made into a full version by training the text or creating a data module.

In order to improve the accuracy of the product, the user feedback can be made use to train the text and thus by time being this product can be evolved in to a more successful product.

**The Aim**: If the user enters a phrase, he should be return with best possible meaning to the phrase.

Example:

>>input: *“a strong positive emotion of regard and affection”*

*>>output:”Love”*

**Sample process:**

1. Tokenize the user input in to nouns and create them in a list.
2. Create a dictionary with words as keys and Nouns of the word’s definition as values.
3. Now compare the similarity between the list of nouns(userinput) and nouns(values)
4. Return the word(key) with max similarity value

**Needed Data packages:**

National language tool kit(NLTK) ,Wordnet.

NLTK Fuctions used: sent\_tokenize, word\_tokenize , pos\_tag

Dictionary for English used: Most common used English words

<http://www.sterlingcrispin.com/100x1000/1000.txt>

Sample words: Barely, bathroom, be, beat, beautiful, became, because, become

POS tag list:

CC coordinating conjunction

CD cardinal digit

DT determiner

EX existential there (like: "there is" ... think of it like "there exists")

FW foreign word

IN preposition/subordinating conjunction

JJ adjective 'big'

JJR adjective, comparative 'bigger'

JJS adjective, superlative 'biggest'

LS list marker 1)

MD modal could, will

NN noun, singular 'desk'

NNS noun plural 'desks'

NNP proper noun, singular 'Harrison'

NNPS proper noun, plural 'Americans'

PDT predeterminer 'all the kids'

POS possessive ending parent's

PRP personal pronoun I, he, she

PRP$ possessive pronoun my, his, hers

RB adverb very, silently,

RBR adverb, comparative better

RBS adverb, superlative best

RP particle give up

TO togo 'to' the store.

UH interjection errrrrrrrm

VB verb, base form take

VBD verb, past tense took

VBG verb, gerund/present participle taking

VBN verb, past participle taken

VBP verb, sing. present, non-3d take

VBZ verb, 3rd person sing. present takes

WDT wh-determiner which

WP wh-pronoun who, what

WP$ possessive wh-pronoun whose

WRB wh-abverb where, when

Functions inside the program:

For returning nouns:

def returnNouns(inputString):

tokens = nltk.word\_tokenize(inputString)

return list(set([word for word, pos in nltk.pos\_tag(tokens) if pos.startswith('NN')]))

The above function will tokenize the phrase takes out the nouns using pos\_tag and store them in a list

For Finding Similarity:

def findSimilarity(word1, word2):

try:

w1 = wordnet.synset(word1 + '.n.01')

w2 = wordnet.synset(word2 + '.n.01')

#print w1, w2

return w1.wup\_similarity(w2)

except:

return 0

print 'error occured'

The above similarity function will calculate the similarity between words. The words are called in a way that .synset would recognize and run though similarity calculations

For comparing the nouns in each list:

def compaire():

#Asking user for the input

userInput = raw\_input('Please enter a phrase which is defination for something: ')

userInputTokenList = returnNouns(userInput)

>> #Here wordDict.json is the processed dictionary which has words as key and nouns of each definition of the key as Value.

>with open('wordDict.json', 'r') as wordDict:

>wordDictData = json.load(wordDict)

>result = 0

>resultKey = ''

>for key in wordDictData.keys():

#Now iterate through Cartetian product of the nouns from user Input and nouns form the wordlist

combinedList = list(itertools.product(wordDictData[key], userInputTokenList))

# Returns list of Tuples

>count = 1

>maxSimilarity = 0

>for tup in combinedList:

>if len(tup) == 2:

#Adding similarity for each of the combination nouns.

>maxSimilarity += findSimilarity(tup[0], tup[1])

>count += 1

#Taking out the avg of the cartetian product

>result = max(maxSimilarity / count, result)

>if result == maxSimilarity / count:

>resultKey = key

#Returning the max similar key

>return (resultKey, result)

print '\nThe closest meaning of the phrase is ' + str(finalResult[0]).upper() + ' with similarity of ' + str(finalResult[1] \* 100)[:5] + '%'

The above function is a combination of sub functions. The above function does the comparing. finding similarity, finding max similar word and of cource cartetian product of nouns in the each list.

I used word net dictionary to define all the meanings to the words. There are few limitations and errors in word net package.

Testing and output:

Successful out comes;

1: Please enter a phrase....”the time after sunset and before sunrise while it is dark outside”

The closest meaning of the phrase is NIGHT with similarity of 62.11%

2. Please enter a phrase....a strong positive emotion of regard and affection

The closest meaning of the phrase is ANGER with similarity of 58.42%

3. Please enter a phrase....an artistic form of auditory communication incorporating instrumental or vocal tones in a structured and continuous manner

The closest meaning of the phrase is DANCE with similarity of 36.80%

The lesser the similarity it shows the more the possible outcomes the phrase has inside the dictionary.

4. Please enter a phrase....a living organism characterized by voluntary movement

The closest meaning of the phrase is ANIMAL with similarity of 43.93%

Not so Accurate results:

1. Please enter a phrase....the most common medium of exchange; functions as legal tender

The closest meaning of the phrase is GRAB with similarity of 27.61%

2. Please enter a phrase....a verbal commitment by one person to another agreeing to do (or not to do) something in the future

The closest meaning of the phrase is SOON with similarity of 35.35%

In the above when tested for Definition of “Money” as input it returned “Grab”

And when entered definition of “Promise” it returned “Grab”  
  
  
Test Conclusion: The accuracy was almost 70 to 30 %. I think its because of, I was using 1000 most common used words. Which also has “a”, “as” etc?

If I can use most 1000 common nouns used, then the accuracy might become better.

They were few errors which I found while testing wordnet.

Inside errors:

import nltk

from nltk.corpus import wordnet

actual=wordnet.synsets('orange')[0]

predicted=wordnet.synsets('orange')[0]

similarity=actual.wup\_similarity(predicted)

print similarity

>>Output 0.75

When two similar words where compared, the similarity of the words were .75 which was supposed to be 1. There is no exact reason why this is happening. But due to this error the chances are accuracy of the output goes down.

Future Scope: If I can compare more POS tags. Then I can compare bigger phrases and get more accurate results.

However, integrating machine learning into the script would make a fine efficient product.

References:

1. [http://pythonprogramming.net](http://pythonprogramming.net/)
2. <https://youtu.be/T68P5-8tM-Y?list=PLQVvvaa0QuDf2JswnfiGkliBInZnIC4HL.html>
3. <https://developmentality.wordpress.com/2012/03/30/three-ways-of-creating-dictionaries-in-python.html>
4. <http://stackoverflow.com/questions/14089887/nltk-pos-tag-usage.html>
5. <http://www.nltk.org/book/ch05.html>