```
# This script identifies the features of the sentences in comparison to the ontology.
# For the moment we have six features, which are also rated individually by users in Airbnb.
The list can/should of course be extended.
# In a sentence we can identify zero or all of these features.
# If only one feature is mentioned, then the sentiment score of the sentence would be 100% (or
1) for this certain feature
# that we found to be mentioned. If there are two mentioned the chances are 50-50 and so on.
# Clearly the sum of probabilities for each sentence every time, must be 1.

#the output shall be something like: (0,1,0,0,1,0) where 0 means the feature is not mentioned
and 1 the opposite

from __future__ import division

def identifier(sentence):
    try:
        # The array named output will store the probabilities of each feature (6 in our case).
        It is initialized with 0.
        output =[0,0,0,0,0,0]

        # Our features as of Airbnb
        features = ['accuracy','check-in','cleanliness','communication','location','value']

        # Count keeps the number of features found. Can be higher than 6 since a feature may be
        mentioned more than once.
        count = 0
        for word in sentence:
            for index,item in enumerate(features):

                # For each feature, read the list of synonyns based on the lemmatized synsets
                of WordNet.
                synonyms = open('C:/Python27/ontology/'+ item + '_noduplicates.txt', 'r')
                lines = synonyms.readlines()

                # If a synonym of the feature is found, increase the index of mentioned feature
                with one and increase the nr of features found.
                for line in lines:
                    if (str(line).replace('\n','') == word):
                        output[index] +=1
                        count +=1
                        break

        # Calculate the probabilities of features' sentiment based on the number of feature
        found in one sentence.
        for i, element in enumerate(output):
            output[i] = round(element/count, 4)
        return output

        # We use try/except so even if errors happen the output will be returned. The most
        frequent error will be
        # the division by 0 in a sentence where no feature is found.
    except:
        return output
```