keywords selection

March 20, 2019

0.0.1 Airbnb's description of all aspects.

- `Overall Experience` What was your guest's overall experience?
- `Cleanliness` Did your guests feel that your space was clean and tidy?
- `Accuracy` How accurately did your listing page represent your space?
- `Value` Did your guest feel your listing provided good value for the price?
- `Communication` How well did you communicate with your guest before and during their stay?
- `Check-in` How smoothly did their check-in go?
- `Location` How did guests feel about your neighborhood?

```
In [1]: # use wordnet find keywords
        from nltk.corpus import wordnet as wn
        import re
```

```
In [2]: # functions for extracting the keywords of an aspect.
        def synset_description(word):
            for synset in wn.synsets(word):
                print("%s : %s" % (synset.name(), synset.definition()))

        def synonym_words_in(word, include_list):
            thesaurus = []
            for synset in wn.synsets(word):
                if synset.name() in include_list:
                    thesaurus += synset.lemma_names()

            return list(set(thesaurus))

        def synonym_words_ex(word, exclude_list):
            thesaurus = []
            for synset in wn.synsets(word):
                if synset.name() not in exclude_list:
                    thesaurus += synset.lemma_names()

            return list(set(thesaurus))

        def anton_list(syn_include_list):
```

```
        anton_list = []
        for synset in syn_include_list:
            for l in wn.synset(synset).lemmas():
                if l.antonyms():
                    anton_list += l.antonyms()

        return anton_list

    def synonyms_of_antonyms(anton_list):
        antonyms = []
        pattern = r"Lemma\('([a-z]+.[a-z].\d+)"

        for anton in anton_list:
            antonyms += wn.synset(re.findall(pattern, str(anton))[0]).lemma_names()

        return list(set(antonyms))
```

1 Determine the keywords relevant to aspect `location`.

The Airbnb officially determine the aspect `location` as: * `Location` How did guests feel about your neighborhood?

So, I will use the two main keywords, `location` and `neighborhood`, to find synonyms for further review text processing.

1.0.1 Firstly, I will look at the synset description of `location`.

This is to remove the unrelevant synset and then increase the accuracy of the set of synonyms.

```
In [3]: synset_description('location')
```

```
location.n.01 : a point or extent in space
placement.n.03 : the act of putting something in a certain place
localization.n.01 : a determination of the place where something is
location.n.04 : a workplace away from a studio at which some or all of a movie may be made
```

1.0.2 Then, exclude not necessarily relevant synset and get synonyms.

From above, we can see the `location.n.04`, `localization.n.01`, `placement.n.03` are not necessarily in accord to the Airbnb's definition of aspect `location`.

So, I will ignore synonyms in those synset.

As a result, the synonyms I will use later will be:

```
In [4]: synw1 = synonym_words_in('location', ['location.n.01'])
```

1.0.3 Do the same thing for word `neighborhood`.

```
In [5]: synset_description('neighborhood')
```

```
vicinity.n.01 : a surrounding or nearby region
neighborhood.n.02 : people living near one another
region.n.04 : the approximate amount of something (usually used prepositionally as in `in the r
neighborhood.n.04 : an area within a city or town that has some distinctive features (especiall
```

```
In [6]: synw2 = synonym_words_ex('neighborhood', ['neighborhood.n.02'])
```

1.0.4 Find antonyms and their synonyms

```
In [7]: # this is a list of synsets that are selected as useful synset in finding synonyms.
        syn_include_list = ['location.n.01', 'vicinity.n.01', 'region.n.04', 'neighborhood.n.04

        # a list of antonym synsets for above list.
        anton_list(syn_include_list)

        # all lemma words in upper antonym synsets.
        antw = synonyms_of_antonyms(anton_list(syn_include_list))
```

1.0.5 Finalize the aspect synonyms set of `location` aspect.

```
In [8]: location_synonyms = list(set(synw1 + synw2 + antw))
        location_synonyms
```

```
Out[8]: ['location',
         'neighbourhood',
         'neighborhood',
         'neck_of_the_woods',
         'region',
         'locality',
         'vicinity']
```

2 Determine the keywords relevant to aspect `cleanliness`.

The Airbnb officially determine the aspect `clealiness` as: * `Cleanliness` Did your guests feel that your space was **clean** and **tidy**.

So, I will use the three main keywords, `clean`, `tidy` and `cleanliness`, to find synonyms for further review text processing.

The process will be the same as `location`.

2.0.1 `cleanliness`

```
In [9]: synset_description('cleanliness')
```

```
cleanliness.n.01 : the habit of keeping free of superficial imperfections
cleanliness.n.02 : diligence in keeping clean
```

```
In [10]: synw1 = synonym_words_ex('cleanliness', '')
```

2.0.2 tidy

```
In [11]: synset_description('tidy')
```

```
tidy.n.01 : receptacle that holds odds and ends (as sewing materials)
tidy.v.01 : put (things or places) in order
tidy.a.01 : marked by order and cleanliness in appearance or habits
kempt.s.01 : (of hair) neat and tidy
goodly.s.01 : large in amount or extent or degree
```

```
In [12]: synw2 = synonym_words_in('tidy', ['tidy.v.01', 'tidy.a.01'])
```

2.0.3 clean

```
In [13]: # synset_description('clean')
```

```
In [14]: synw3 = synonym_words_in('clean', ['clean.v.01', 'houseclean.v.01', 'clean.v.05', 'cl
                                            'scavenge.v.04', 'clean.a.01'])
```

2.0.4 Find antonyms and their synonyms

After I used the synonyms as keywords to extract sentences that are relevant to `cleanliness` aspect from all review text, I found that the accuracy is pretty high.

However, it neglects some sentences that used the antonyms of the keywords like "the kitchen needed a good scrubbing - it was **dirty**".

So, I decided to also add antonyms and their synonyms into keywords list using the `anton_list` and `synonyms_of_antonyms` function.

```
In [15]: # this is a list of synsets that are selected as useful synset in finding synonyms.
         syn_include_list = ['cleanliness.n.01', 'cleanliness.n.02', 'tidy.v.01', 'tidy.a.01',
                             'clean.v.01', 'houseclean.v.01', 'clean.v.05', 'clean.v.08',
                             'scavenge.v.04', 'clean.a.01']
```

```
In [16]: # a list of antonym synsets for above list.
         anton_list(syn_include_list)
```

```
Out[16]: [Lemma('uncleanliness.n.01.uncleanliness'),
          Lemma('untidy.a.01.untidy'),
          Lemma('dirty.v.01.dirty'),
          Lemma('dirty.a.01.dirty')]
```

```
In [17]: # all lemma words in upper antonym synsets.
         antw = synonyms_of_antonyms(anton_list(syn_include_list))
```

2.0.5 Finalize the aspect synonyms set of `cleanliness` aspect.

```
In [18]: cleanliness_synonyms = list(set(synw1 + synw2 + synw3 + antw))
         cleanliness_synonyms
```

```
Out[18]: ['colly',
          'neaten',
          'straighten',
          'soil',
          'tidy',
          'make_clean',
          'tidy_up',
          'clean',
          'begrime',
          'straighten_out',
          'bemire',
          'square_away',
          'unclean',
          'clean_up',
          'untidy',
          'grime',
          'dirty',
          'cleanliness',
          'scavenge',
          'soiled',
          'clean_house',
          'houseclean',
          'uncleanliness']
```

```
In [200]: # #
          # wn.synset('goodly.s.01').hypernyms()
          # #
          # wn.synset('cleanliness.n.02').lemmas()[0].antonyms()

          # def word_similarity(w1, w2):
          #     token1 = nlp(w1)
          #     token2 = nlp(w2)
          #     return token1.similarity(token2)

          # antonym_words('clean.a.01')
```