Lab assignment: Aspect Opinion extraction

Sáez Maldonado Fco Javier, Álvarez Ocete José Antonio

March 31, 2022

Introduction

In this document, we will comment all the tasks and sub tasks that we have coded for this assignment. The most relevant pieces of code and results will be exposed in the following sections.

0 Tasks accomplished

We provided a list of all the accomplished tasks:

- **Task 1.1** - mandatory
- **Task 1.2** - optional
- **Task 1.3** - optional
- **Task 2.1** - mandatory
- **Task 2.2** - optional
- **Task 2.3** - optional
- **Task 3.1** - mandatory
- **Task 3.2** - optional
- **Task 3.3** - optional (partially accomplished)
- **Task 4.1** - mandatory
- **Task 4.2** - optional
- **Task 4.3** - optional (partially accomplished)
- **Task 5.1** - mandatory
- **Task 5.2** - mandatory

In this document we focus on tasks 4 and onwards, since they are the most interesting ones. The code and examples from previous tasks may be found in the associated notebook.

1 Assignment 1

1.1 Task 1.1

Loading all the hotel reviews from the Yelp hotel reviews file.

1.2 Task 1.2 (optional)

Loading line by line the reviews from the Yelp beauty/spa resorts and restaurants reviews files.

1.3 Task 1.3 (optional)

Loading line by line reviews on other domains (e.g., movies, books, phones, digital music, CDs and videogames) from McAuley's Amazon dataset.

We tackle all of these tasks at the same time since a general enough functions solves all of them directly. The function `load_json_line_by_line()` reads a json file line by line and returns the dataset built.

We additionally created a test function that tests the loading of all the described datasets. We have selected the Amazon cell phones and accessories dataset because it is big enough without being huge and we also have the required aspects for it (see tasks 2.1 to 2.3). In this memory, we only include an example, the rest can be found in the jupyter notebook.

```
Reading file inputs/yelp_dataset/yelp_hotels.json

5034 reviews loaded

Example review: {'reviewerID': 'qLCpuCWCyPb4G2vN-WZz-Q', 'asin':
'8ZwO9VuLDWJOXmtAdc7LXQ', 'summary': 'summary', 'reviewText': "Great hotel in
Central Phoenix for a stay-cation, but not necessarily a place to stay out of
town and without a car. Not much around the area, and unless you're familiar
with downtown, I would rather have a guest stay in Old Town Scottsdale, etc.␣
↪BUT
if you do stay here, it's awesome. Great boutique rooms. Awesome pool that's
happening in the summer. A GREAT rooftop patio bar, and a very very busy lobby
with Gallo Blanco attached. A great place to stay, but have a car!", 'overall':
4.0}
```

2 Assignment 2

2.1 Task 2.1

Loading (and printing on screen) the vocabulary of the `aspects_hotels.csv` file, and directly using it to identify aspect references in the reviews. In particular, the aspects terms could be mapped by exact matching with nouns appearing in the reviews.

We will compute a dictionary that matches a certain aspect to every word related to it. It will usually be called `aspect_words_dict`. This will optimize knowing which aspect is related to each word.

The function build_simple_vocab creates this dictionary given a path to the file with the initial vocabulary. We introduce it in a dataframe (used only with displayability purposes) and we display the result.

[3]:

	0
amenity	amenities
amenities	amenities
services	amenities
atmosphere	atmosphere
atmospheres	atmosphere
ambiance	atmosphere
ambiances	atmosphere
light	atmosphere
lighting	atmosphere
lights	atmosphere

In the following cells we compute the aspects referenced by each review and display the result for the first few reviews.

```
Review: Great hotel in Central Phoenix for a stay-cation, but not
necessarily a place to stay out of town and without a car. Not much around the
area, and unless you're familiar with downtown, I would rather have a guest␣
↪stay
in Old Town Scottsdale, etc. BUT if you do stay here, it's awesome. Great
boutique rooms. Awesome pool that's happening in the summer. A GREAT rooftop
patio bar, and a very very busy lobby with Gallo Blanco attached. A great place
to stay, but have a car!
Aspects: {'bar', 'shopping', 'building', 'pool', 'transportation'}

Review: I feel the Days Inn Tempe is best described as "a place where
you can purchase the right to sleep for awhile." I booked my 10-night stay on
Travelocity for a non-smoking room, yet when I entered the room I almost␣
↪choked.
It was disgusting. I've never had a smoking hotel room before and I will make
sure I don't again. They said they couldn't move us to a different room.My␣
↪local
lady friend brought over a bottle of wine but forgot a corkscrew. No big deal,␣
↪I
thought to myself, as the front desk of a hotel will surely have a corkscrew.
Nope. Coors Light it is.The towels felt like they were made of cow tongue, and
they missed our wakeup call one morning making me late for a training class␣
↪that
had cost me about $500.I'm awarding one star in addition to the minimum␣
↪one-star
rating because I got to drink cheap beer by the pool in 90 degree weather for␣
↪10
days, and there are a few good places to eat and two dollar stores very nearby.
```

```
The dollar store had a corkscrew.
Aspects: {'atmosphere', 'bedrooms', 'drinks', 'breakfast', 'shopping',
'bathrooms', 'price', 'pool'}
```

2.2 Task 2.2 (optional)

Generating or extending the lists of terms of each aspect with synonyms extracted from WordNet.

For this second task we expand the vocabulary using synonims extracted from Wordnet. The function `build_vocab()` is analogous to the previous `build_simple_vocab()` but takes this synonims into account.

2.3 Task 2.3 (optional)

Managing vocabularies for additional Yelp or Amazon domains. See assignments 1.2 and 1.3

Extended our previous functions to the new datasets is trivial. We simple need to load the correct aspects for each review. The following test function computes the following for the Yelp hotels, Yelp restaurants and Amazon phones datasets:

- Load the reviews and build both the simple and complex vocabularies.
- Print the aspects found in the first few reviews with each vocabulary.
- Print the number of words in both the simple and extended vocabulary for comparison.

```
Review: Great hotel in Central Phoenix for a stay-cation, but not
necessarily a place to stay out of town and without a car. Not much around the
area, and unless you're familiar with downtown, I would rather have a guest␣
↪stay
in Old Town Scottsdale, etc. BUT if you do stay here, it's awesome. Great
boutique rooms. Awesome pool that's happening in the summer. A GREAT rooftop
patio bar, and a very very busy lobby with Gallo Blanco attached. A great place
to stay, but have a car!
Aspects: {'bar', 'shopping', 'building', 'pool', 'transportation'}
Extended Vocab Aspects: {'bedrooms', 'pool', 'bar', 'service',
'location', 'shopping', 'checking', 'building', 'transportation'}
```

From this point onwards we will focus on the hotel dataset. The hotel reviews and vocab are loaded now and fixed for the rest of this practical assignment.

3 Assignment 3

3.1 Task 3.1 (mandatory)

Loading Liu's opinion lexicon composed of positive and negative words, accessible as an NLKT corpus, and exploiting it to assign the polarity values to aspect opinions in assignment 4.

We decided to load not only Liu's lexicon but also 'Vader' lexicon. Also, we used 'SentiWordNet', but we do not need to load it in advance to its usage, so we directly use it in the code.

3.2 Task 3.2 (optional)

Considering modifiers to adjust the polarity values of the aspect opinions in Assignment 4.

These are loaded from the provided csv, and used in assginment 4.

3.3 Task 3.3 (optional)

Considering negation of opinion words and negation of sentences to adjust the polarity values of the aspect opinions in Assignment 4.

This task has been partially accomplished since we have only taken into account negations on verbs (like *isn't* or *didn't*), see the defined grammar below. We could have somply added another negation is a *not* word was present in the sentence, but we felt that some cases didn't fit at all with this implementation, like the common use *noy only [...] but also [...]*.

4 Assignment 4

Once the aspect vocabulary and opinion lexicons are loaded, the opinions about aspects have to be extracted from the reviews. For this purpose, POS tagging, constituency and dependency parsing could be used. - POS tagging would allow identifying the adjectives in the sentences. - Constituency and dependency parsing would allow extracting the relations between nouns and adjectives and adverbs.

The following tasks are proposed:

- **Task 4.1 (mandatory)**: extracting the `[aspect, aspect term, opinion word, polarity]` tuples from the input reviews
- **Task 4.2 (optional)**: extracting the `[aspect, aspect term, opinion word, modifier, polarity]` tuples from the input reviews, taking the modifiers of assignment 3.2
- **Task 4.3 (optional)**: extracting the `[aspect, aspect term, opinion word, isNegated, polarity]` tuples from the input reviews, taking the modifiers of assignment 3.3

We firstly create code to extract the asked tuples. This is no more than joining (with a little bit of care) all the code that we have previously developed. We also parts of the code provided in the assignment. We begin by creating a function that returns the POS tagging for a given text, and we test it.

[14]:
```
print(pos_tagging("I think you are very handsome."))
```

```
[[('I', 'PRP'), ('think', 'VBP'), ('you', 'PRP'), ('are', 'VBP'), ('very',
        'RB'), ('handsome', 'JJ'), ('.', '.')]]
```

Exploring a few examples, we found the following **difficulty**: some adjectives were not detected as adjectives since the first letter was uppercased. We decided to **lowercase** the review text. It could be interesting to see if more preprocessing to the text could improve the results.

Note.- A list of possible returned POS tags returned by NLTK can be found here.

We define a grammar that tries to capture:

- (Adverb) + Adjective + Noun
- Noun + Verb + (Adverb) + Adjective

In particular, there might be more of each element inthe respective position (i.e., two adjectives, two adeverbs, etc.). We tested this in very simple examples.

[16]:
```
# GRAMMAR TESTING
grammar = r"""
JJNN: {<RB.*>*<JJ.*>+<NN.*>+} # adjectives escorting nouns
      {<NN.*>+<VB.*><RB.*>*<JJ.*>+} # description sentences
"""
```

```
Sentence:  The place is perfect
    [('location', 'place', 'perfect', 'not negated', '', 2.7)]

Sentence:  The place is absolutely perfect
    [('location', 'place', 'perfect', 'not negated', 'absolutely', 5.4)]

Sentence:  The place isn't perfect
    [('location', 'place', 'perfect', 'negated', '', -2.7)]

Sentence:  The place isn't absolutely perfect
    []
```

We can appreciate that the grammar is working well in all the proposed tests but in the last one. In this case, our grammar is not capturing the adjective "perfect" as an adjective. Let us see what `pos_tagging` returns:

[17]:
```
pos_tagging("The place isn\'t absolutely perfect")
```

[17]:
```
[[('The', 'DT'),
        ('place', 'NN'),
        ('is', 'VBZ'),
        ("n't", 'RB'),
        ('absolutely', 'RB'),
        ('perfect', 'VB')]]
```

As we can see, the problem is that the *POS Tagger* indicates that `perfect` is a verb. In this case, this is obviously not true, so it would be very interesting to use a different tagger in future projects.

We now test the proposed code in the hotel reviews:

[18]:
```
def test_aspect_opinions(first_n=3, grammar=grammar):

for review in hotel_reviews[:first_n]:
review_text = review['reviewText']
print('Review text: ', review_text, '\n')
pprint(aspect_opinions_from_review(review_text, word_aspect_dict,
adj_polarities, modifiers, grammar, method='SentiWordNet'))
```

```
    print('')


    test_aspect_opinions()
```

```
Review text:  Great hotel in Central Phoenix for a stay-cation, but not
necessarily a place to stay out of town and without a car. Not much around the
area, and unless you're familiar with downtown, I would rather have a guest␣
↪stay
in Old Town Scottsdale, etc. BUT if you do stay here, it's awesome. Great
boutique rooms. Awesome pool that's happening in the summer. A GREAT rooftop
patio bar, and a very very busy lobby with Gallo Blanco attached. A great place
to stay, but have a car!

[('shopping', 'boutique', 'great', 'not negated', '', 0.0),
('bedrooms', 'rooms', 'great', 'not negated', '', 0.0),
('pool', 'pool', 'awesome', 'not negated', '', 0.75),
('building', 'patio', 'great', 'not negated', '', 0.0),
('bar', 'bar', 'great', 'not negated', '', 0.0),
('building', 'lobby', 'busy', 'not negated', 'very', 0.75),
('building', 'lobby', 'busy', 'not negated', 'very', 0.75),
('location', 'place', 'great', 'not negated', '', 0.0)]

Review text:  I feel the Days Inn Tempe is best described as "a place where you
can purchase the right to sleep for awhile." I booked my 10-night stay on
Travelocity for a non-smoking room, yet when I entered the room I almost␣
↪choked.
It was disgusting. I've never had a smoking hotel room before and I will make
sure I don't again. They said they couldn't move us to a different room.My␣
↪local
lady friend brought over a bottle of wine but forgot a corkscrew. No big deal,␣
↪I
thought to myself, as the front desk of a hotel will surely have a corkscrew.
Nope. Coors Light it is.The towels felt like they were made of cow tongue, and
they missed our wakeup call one morning making me late for a training class␣
↪that
had cost me about $500.I'm awarding one star in addition to the minimum␣
↪one-star
rating because I got to drink cheap beer by the pool in 90 degree weather for␣
↪10
days, and there are a few good places to eat and two dollar stores very nearby.
The dollar store had a corkscrew.

    [('checking', 'stay', '10-night', 'not negated', '', 0.0),
        ('location', 'deal', 'big', 'not negated', '', 0.125),
        ('bathrooms', 'towels', 'is.the', 'not negated', '', 0.0),
        ('drinks', 'beer', 'cheap', 'not negated', '', -0.25)]
```

In this case, we can again observe the problems that the POS tagger causes. In the second shown opinion, we see that **non adjective words are detected as adjectives**. Furthermore, *sequences of characters* ("is.the") that are not even words are marked as adjectives. Clearly, the *non-words* problem could be solved with **text-preprocessing**, but we would need expert information to be able to determine which kind of preprocessing should be done. Also, this would not fix the problem of the POS tagger tagging incorrectly the words.

5 Assignment 5

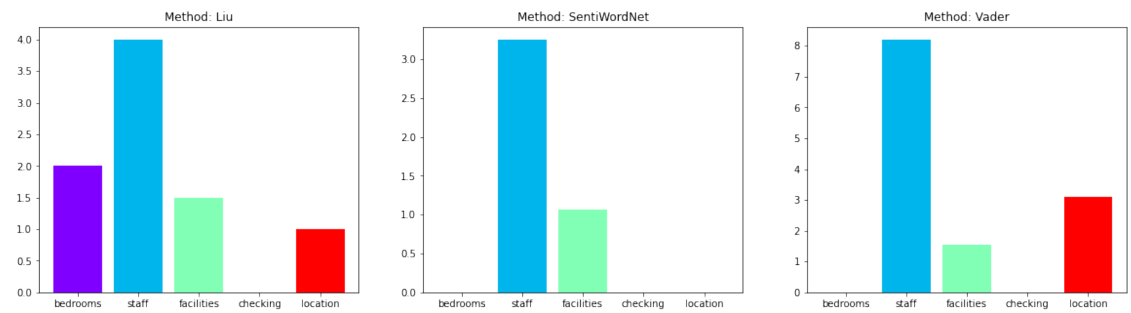
To validate and evaluate the solutions implemented in previous tasks, you are finally proposed the following tasks:

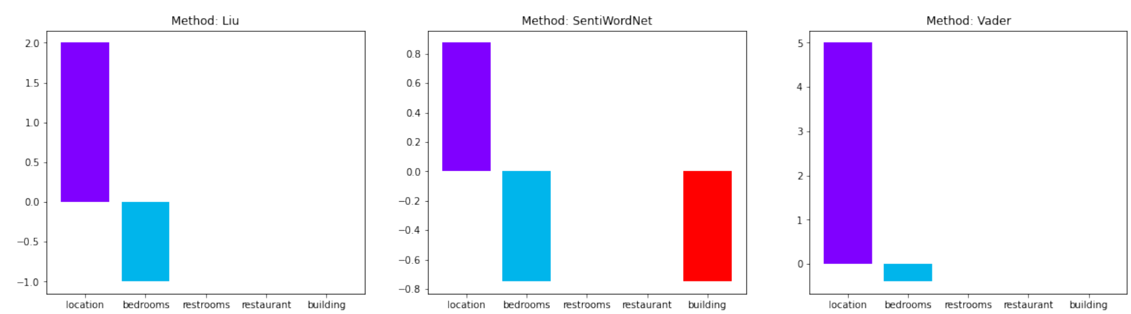
5.1 Task 5.1 (mandatory)

Visualizing on screen the aspect opinions (tuples) of a given review.

We developed a function that summarizes (by summing up) all the aspect opinions for a single review, and barplot the results. We display three different graphs for the different lexicons used.

[20]:
```
plot_review_aspects(hotel_reviews[2]['reviewText'])
plot_review_aspects(hotel_reviews[40]['reviewText'])
```





5.2 Task 5.2 (optional)

Visualizing on screen a summary of the aspect opinions of a given item. Among other issues, the total number of positive/negative opinions for each aspect of the item could be visualized.

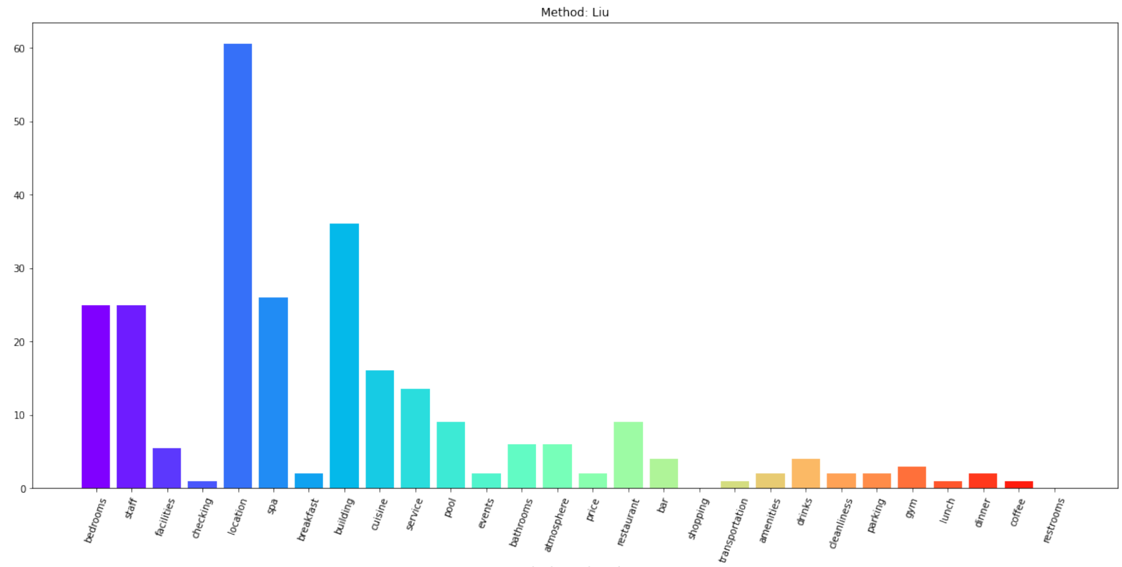
Two types of visualizations are proposed.

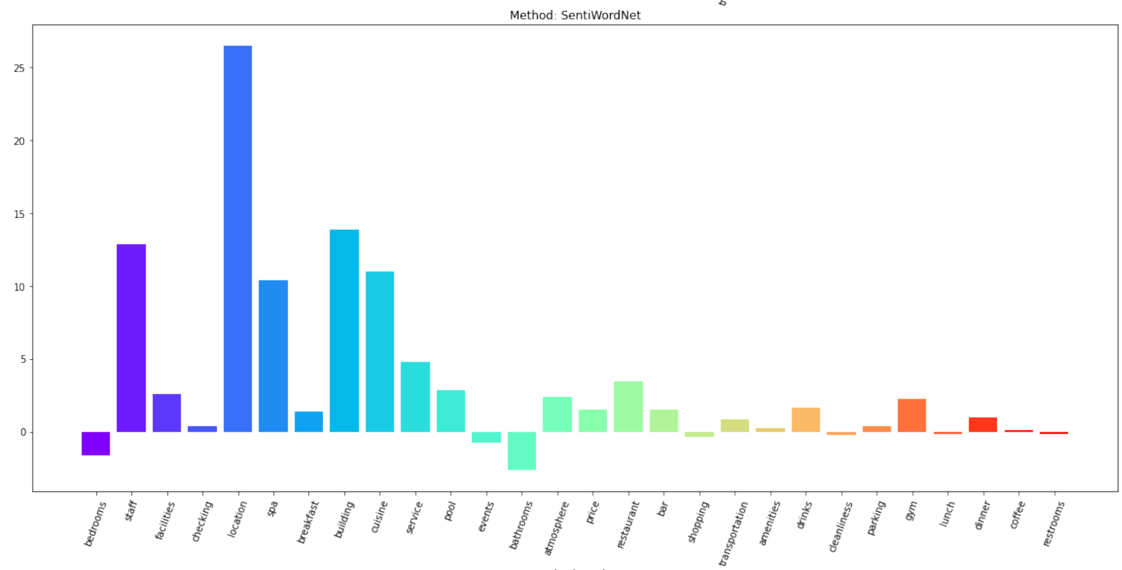
Complete Hotel Plot The first one follows the previous approach we plot the aspect and polarities in a single review. What we do is to **sum** the polarities per aspect in each of the reviews to plot all the extracted aspects and polarities for a single hotel, given by its `asin`, which is an identifier.

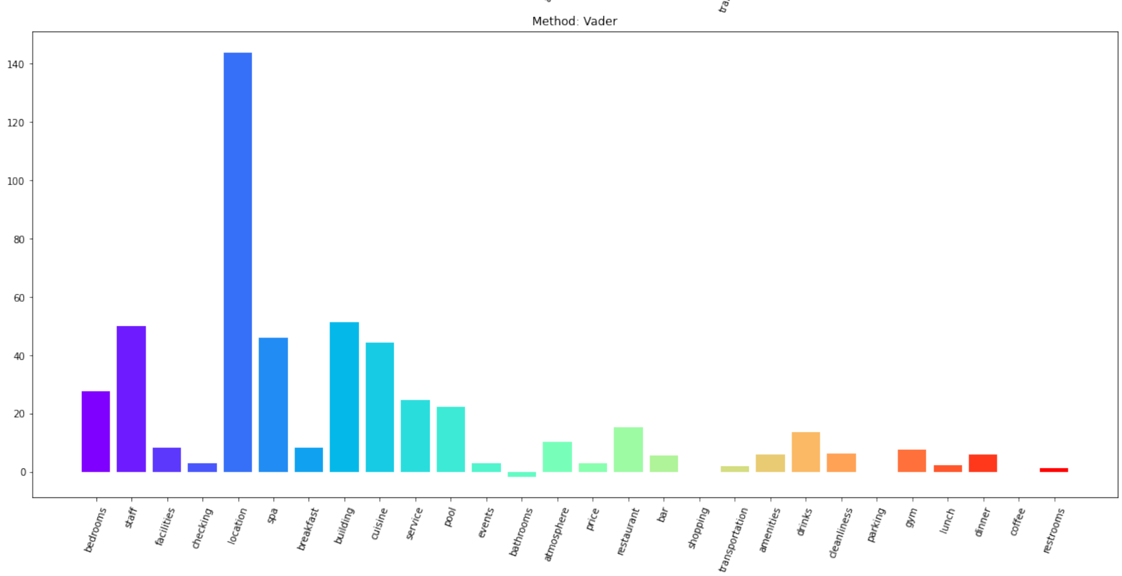
[22]:
```
# Fixed an asin (the ID associated to a hotel)
fixed_asin = hotel_reviews[2]['asin']
fixed_hotel_reviews = [ x for x in hotel_reviews if x['asin'] ==␣
↪fixed_asin ]
print("Selected {} opinions for this hotel".
↪format(len(fixed_hotel_reviews)))

aspects_per_opinion = plot_hotel_aspects(fixed_hotel_reviews)
```

```
Selected 102 opinions for this hotel
```


Method: Liu
0
10
20
30
40
50
60
bedrooms
staff
facilities
checking
location
spa
breakfast
building
cuisine
service
pool
events
bathrooms
atmosphere
price
restaurant
bar
shopping
transportation
amenities
drinks
cleanliness
parking
gym
lunch
dinner
coffee
restrooms

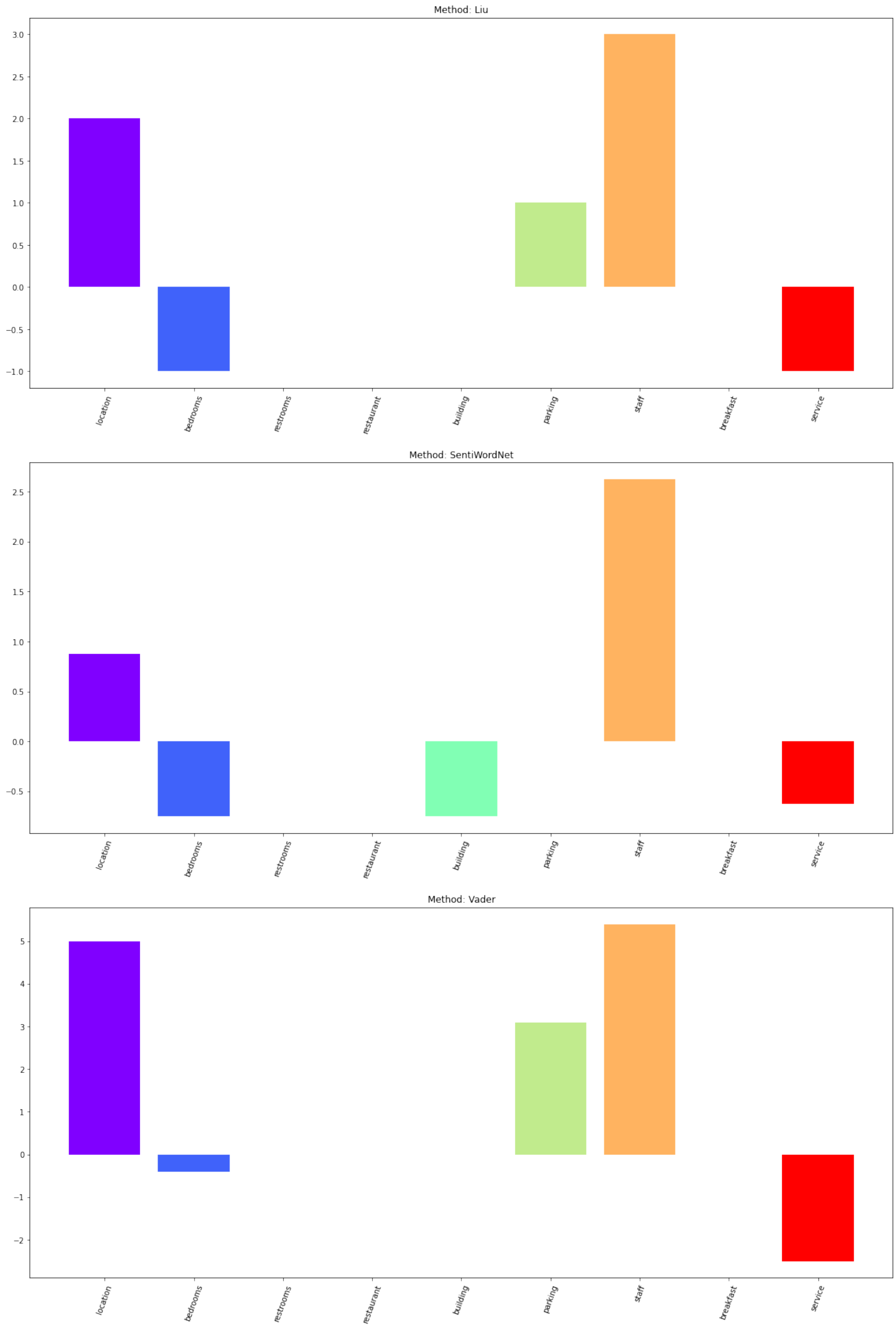

Method: SentiWordNet
0
5
10
15
20
25
bedrooms
staff
facilities
checking
location
spa
breakfast
building
cuisine
service
pool
events
bathrooms
atmosphere
price
restaurant
bar
shopping
transportation
amenities
drinks
cleanliness
parking
gym
lunch
dinner
coffee
restrooms


Method: Vader
0
20
40
60
80
100
120
140
bedrooms
staff
facilities
checking
location
spa
breakfast
building
cuisine
service
pool
events
bathrooms
atmosphere
price
restaurant
bar
shopping
transportation
amenities
drinks
cleanliness
parking
gym
lunch
dinner
coffee
restrooms

As we can see, the opinions on this hotel agree in the location of the hotel. We can see the little differences between the polarity methods, where in the `SentiWordNet` method the negatives seem to have a bigger impact on the final bar plot.

Let us see another hotel with a smaller number of reviews.

```
Selected 5 opinions for this hotel
```


Method: Liu
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
location
bedrooms
restrooms
restaurant
building
parking
staff
breakfast
service
Method: SentiWordNet
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
location
bedrooms
restrooms
restaurant
building
parking
staff
breakfast
service
Method: Vader
5
4
3
2
1
0
−1
−2
location
bedrooms
restrooms
restaurant
building
parking
staff
breakfast
service

In this case, we can see a difference in the polarities in the aspect `building`, since SentiWordNet assigns a lower polarity in any of the reviews.

Numerical statistics about the reviews The last case would be to design a code that is capable to **summarize** all the positive and negative opinions about an aspect given a certain Hotel. The code developed returns the, for **each** of the polarity assignment methods, the following values:

- `single_positive`, indicating the total number of reviews that say **at least** one positive thing about the aspect.
- `single_negative`, the same as the previous one but saying negative things.
- `total_positive` and `total_negative`, which is the sum of total positive/negative things said about that aspect
- `average_positive/negative` the average polarity of all positive and negative comments done.

We can see the output for both previous case, using the one with lesser number of reviews first:

[25]:
```
aspects = ['location','bedrooms', 'service']
hotel_name = fixed_asin = hotel_reviews[40]['asin']
counts = count_aspects_opinion(aspects_per_opinion_2, aspects)
show_statistics_per_aspect(hotel_name, aspects, counts)
```

```
Statistics found for hotel: CYMG5AsrhkhUPro2c6NSUA
Aspect: location
                    Liu  SentiWordNet  Vader
single_positive   1.0          1.000    1.0
single_negative   0.0          0.000    0.0
total_positive    2.0          1.000    2.0
total_negative    0.0          0.000    0.0
average_positive  1.0          0.875    2.5
average_negative  0.0          0.000    0.0

Aspect: bedrooms
                    Liu  SentiWordNet  Vader
single_positive   0.0           0.00    0.0
single_negative   1.0           1.00    1.0
total_positive    0.0           0.00    0.0
total_negative    1.0           1.00    1.0
average_positive  0.0           0.00    0.0
average_negative -1.0          -0.75   -0.4

Aspect: service
                    Liu  SentiWordNet  Vader
single_positive   0.0          0.000    0.0
single_negative   1.0          1.000    1.0
total_positive    0.0          0.000    0.0
total_negative    1.0          1.000    1.0
average_positive  0.0          0.000    0.0
```

```
average_negative -1.0        -0.625   -2.5
```

We can see how the service has the highest value of negative polarity, as we saw in the charts.

In the first case analyzed before, we obtain the following result:

[26]:
```
hotel_name = fixed_asin = hotel_reviews[2]['asin']
counts = count_aspects_opinion(aspects_per_opinion, aspects)
show_statistics_per_aspect(hotel_name, aspects, counts)
```

```
Statistics found for hotel: EcHuaHD9IcoPEWNsU8vDTw
Aspect: location
                        Liu  SentiWordNet      Vader
single_positive   36.000000     29.000000  33.000000
single_negative    3.000000      4.000000   3.000000
total_positive    65.000000     53.000000  57.000000
total_negative     4.000000      5.000000   4.000000
average_positive   1.146154      0.650943   3.042982
average_negative  -1.000000     -0.600000  -2.000000

Aspect: bedrooms
                        Liu  SentiWordNet      Vader
single_positive   20.000000     15.000000  13.000000
single_negative    1.000000     12.000000   1.000000
total_positive    23.000000     15.000000  14.000000
total_negative     1.000000     15.000000   1.000000
average_positive   1.130435      0.433333   2.135714
average_negative  -1.000000     -0.541667  -2.300000

Aspect: service
                        Liu  SentiWordNet      Vader
single_positive   16.000000      10.00000  12.000000
single_negative    4.000000       4.00000   2.000000
total_positive    37.000000      25.00000  23.000000
total_negative    10.000000       8.00000   4.000000
average_positive   0.986486       0.61250   2.656522
average_negative  -1.200000      -0.84375  -3.150000
```

6 Conclusion

After all this work, a few conclusions have been extracted:

1. Defining a complete grammar is complicated. In natural language, there are many possible cases of use of our words, so most of the time it is impossible to capture them all in a grammar. We decided to use a simple grammar for simplicity of analyzing our problems.
2. The POS tagger determines the quality of our analysis, since sometimes it captures wrong tags for certain words, which causes problems with the posterior analysis.

3. Also, the final polarity value depends on the used lexicon. Although sometimes the values are coincident, most of the times the polarity values vary between the lexicons, making our method less robust to a change of lexicon.

In order to outperform our implementation we could tackle either one of the previously mentioned points, or change our point of view and approach this problem using advanced state-of-the-art techniques, such as BERT, as done in this work.