

Project Work in Data Science for Business

Dr. Stephan Huber¹ (Module Coordinator)
your name (Student) Michael Jackson (Student)
Bob Ross (Student)

June 7, 2020

¹stephan.huber@hs-fresenius.de

Preface

This document should help to describes and paraphrases how the project work in the course *Data Science for Business* should look like. The paper is structured as follows: Chapter 1 gives an idea of how the project paper can be structured. In chapter 2 the titles of the projects are listed and chapter 3 offers a brief outlook.

Overall, the idea of the project is that we write a textbook together that can work as an applied introduction to data science for students of business administration. Each project consists of one chapter. Of course, students can work together, form groups, and divide work with respect their preferences and talents, respectively. However, every student is responsible (and is graded) for his/her chapter and his/her presentation only.

Moreover, I will implement (and monitor) your working process using GitHub. I will explain you how that works in a video.

Please notice, that this document is not written to be self-explaining. It should rather work as a template. I explain everything in further detail during a GoToMeeting.

Contents

1	A Sketch of How a Project Should Look Like	6
	BARBARA STREISAND BARBARA@EMAIL.DE	
1.1	Motivation	6
1.2	Methodological Issues	7
1.3	Applications in R	7
1.4	Example(s) From the Real World	7
1.5	Conclusion	7
1.6	Exercises	8
1.7	Test Questions	8
1.8	Glossary	8
2	List of Topics	9
	MICHAEL JACKSON MJ@KING.COM	
3	Further Procedure	10
	BOB ROSS GOODNIGHT@PAINT.ORG	
3.1	L ^A T _E X	10
3.2	Git and GitHub	11
3.3	Style of Writing	11
4	Sentiment analysis	12
	RYAN ZIDAGO ZIDAGO.RYAN@HS-FRESENIUS.DE	
4.1	Motivation	13
4.2	Methodological Issues	14
4.3	Application in Python	16
4.4	Example(s) From the Real World	23
4.5	Conclusion	24
4.6	Exercises	25

4.7	Test Questions	26
4.8	Glossary	27

List of Tables

3.1	This is a table	11
-----	---------------------------	----

List of Figures

3.1	This is a picture	11
-----	-----------------------------	----

Chapter 1

A Sketch of How a Project Should Look Like

BARBARA STREISAND

BARBARA@EMAIL.DE

Summary This chapter offers a sketch on how a project may be structured. A project should include a summary, learning objectives, a motivation, a methodological section, a section with an application in \mathbb{R} , an example from businesses, a conclusion, exercises, test questions, and a glossary.

An abstract should make clear in simple language what the chapter is all about. Its maximum length is 200 words.

Learning Objectives

- ;weruhkltewiouthb
- swdfgwserg

1.1 Motivation




Motivate your topic. Discuss why the topic and the content of your chapter may be of importance for the reader. Make the content of your chapter subject of a discussion. Outline clearly what the reader can expect. Include a description of contents. For example, write at the end of this section “The remainder of the chapter is structured as follows: Section 1.2 introduces the

theory...”. Additionally, mention which related topics will not be discussed. Recommend literature for self-study here.¹

1.2 Methodological Issues

Bring the statistical theory or econometric approach to the reader. Try to raise a basic understanding of the meaning and the difficulties of the respective method.

1.3 Applications in R

Explain how your topic can be addressed by using  and RStudio, respectively. In particular, introduce -packages that allow to implement the respective methods. Offer a -script that allows the reader to understand both the method and the programming skills that are needed to apply the method. It should be possible for the reader to replicate the stuff that is presented in the section and the script.

1.4 Example(s) From the Real World

Make an impressive² example from the *real world* to deepen the application and the methods discussed above. The example can stem either from the *academic world* or the *business world*. For example, pick an *academic* research paper and summarize it to the reader in a way that he can understand the strengths and weaknesses of the respective investigation. Or, you can explain a business case where the method of interest plays an important role to solve a problem or to earn money, for example.

Sections 4.3 and 4.4 can be combined.

1.5 Conclusion

Conclude the chapter. This may include a short summary, an outlook and/or related literature.

¹Making a footnote or citing in L^AT_EX is easy. For example, Provost and Fawcett (see 2013, p. XY) or (Provost and Fawcett, 2013)

²The example should be a significant academic contribution, i.e., the example is published in a highly ranked academic journal. Alternatively, the economic impact of the example should be sizable.

1.6 Exercises

- (1) Design an exercise that helps the reader to understand and repeat important concepts.
 - a) Challenge the reader to think about the topic.
 - b) Encourage a discussion.
 - c) Bring up new aspects of the method.
 - d) Be creative.

1.7 Test Questions

- Ask five short test questions that refer to five important insights from your chapter. A discerning reader should have no problems with these questions.

1.8 Glossary

word This text should define the meaning of word.

Chapter 2

List of Topics

MICHAEL JACKSON
MJ@KING.COM

Summary Read the following titles of potential projects and send me an email with three topics you could imagine working on (stephan.huber@hs-fresenius.de). Rank them according to your preferences. I will try to consider your wishes. Feel free to make your own suggestions. But please, don't forget to name overall three (!) topics. If everybody likes to have the same topic, I flip a coin and/or try to consider your second and third preference.

1. Data Collection in Data Science
2. Exploratory Data Analysis in Data Science
3. Regression Analysis in Data Science
4. Spatial Analysis in Data Science
5. Nearest Neighbor Analysis in Data Science
6. Decision Trees Analysis in Data Science
7. Text Mining Analysis in Data Science
8. Big Data Analysis in Data Science
9. Machine Learning in Data Science
10. Single-Board Computers in Data Science

Chapter 3

Further Procedure

BOB ROSS
GOODNIGHT@PAINT.ORG

The project work includes the following:

- Write a chapter on you topic as paraphrased above. Length of the paper should be about 25 pages.
- Prepare a presentation of 20 minutes length (we will see how we manage that part).
- Use the Latex to write the chapter.
- Use Github to include your chapter into the book which contains all chapters.
- Use R and upload your R-code to Github.

3.1 \LaTeX

\LaTeX is easy and powerful. It allows the author to focus on the content because he don't need to take care about the layout very much. There are millions of sources online that provide tutorials. You can include figures in an floating environment¹, see Figure 3.1. Tables are possible, too. See Table 3.1 or you can reference online sources, see <https://texdoc.net/texmf-dist/doc/latex/booktabs/booktabs.pdf> for a guide to make nice tables.

¹The placement of the figure is optimized automatically.



Figure 3.1: This is a picture

Table 3.1: This is a table

Item		
Animal	Description	Price (\$)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

3.2 Git and GitHub

GitHub provides hosting for software development version control using Git which is a version control system designed to handle projects with many contributors. Both tools are heavily used in software engineering and data science. It is particularly powerful when teams work on projects with a procedural workflow.

3.3 Style of Writing

Assume your reader is a well informed master student of business administration. Make things easy for your reader. Believe me, this sounds so easy but it is actually the most difficult task for scientists and academic writers. I got a lot out of reading *Writing Tips for Ph. D. Students* from Cochrane (2005)². While you are not a Ph. D. student, these writing tips apply for all authors who aim to communicate efficiently.

²You can download this file using <https://t1p.de/br5o>

Chapter 4

Sentiment analysis

RYAN ZIDAGO

ZIDAGO.RYAN@HS-FRESENIUS.DE

Summary Sentiment analysis has seen a growing interest in the last few years in data science, particularly due to user-generated content (UGC) becoming more ubiquitous than ever on the modern Web 2.0. As such, companies look out for efficient ways to leverage vast amounts of UGC to assess their reputation, as well as improving their current products and services, based on continuous online customer feedback. One way to proceed is by the mean of sentiment analysis, a subset of text mining mostly concerned with sentiments and opinions that are contained in texts. In the following chapter, the reader will be introduced to one of the most practical ways to conduct such an analysis, namely the dictionary-based approach.

Learning Objectives

- You will get a basic understanding of sentiment analysis and its real-world application
- You will understand the different approaches to sentiment analysis and their trade-offs
- You will learn how to navigate the open-source ecosystem to find flaws or information in external software dependencies
- You will conduct simple dictionary-based sentiment analysis in an independent manner with the aid of the Python programming language and the vaderSentiment library

- You will critically reflect on the produced work and assess its limitations as well as its possibilities for further improvement

4.1 Motivation

Natural languages (like English, German or French), are highly complex constructs. Humans are efficient at dealing with ambiguity and interpret irony, sarcasm, humor, figure of speech, double meaning, implicitness, innuendos and so forth. Machines are best at following a set of predefined rules, executing repetitive tasks concurrently and at a great pace, during long period of time.

Sentiment analysis is at the crossroad between humans and machines; it leverages both of them to figure out what humans really mean when they voice their opinions, concern or appreciations on the public sphere that is the Internet. On one side, we have humans that built dictionaries full of tokens and their associated sentiment, as interpreted by themselves, and on the other side, we have machines that can compute the sentiment of millions of texts in a comparatively to human, extremely short amount of time.

There are numerous applications to sentiment analysis: it is widely used in marketing, to better understand how consumer describe their experiences online (Xiang, Schwartz, Gerdes and Uysal, 2015), to asses corporate reputation (O'Connor, 2010, Vidya, Fanany and Budi, 2015, Chung, Chong, Chua and Na, 2019), but also in finance, as a mean to predict stock market movements (Mohan, Mullapudi, Sammeta, Vijayvergia and Anastasiu, 2019) and in health care to understand the current mental health of forum users (Davcheva, Adam and Benlian, 2019).

By the end of this chapter, the reader will have a firm understanding of sentiment analysis, the different approaches used, and how to compute sentiment of textual data, with the intent to extract relevant information for decision-making and solving real-world complex business problems.

In the **Methodological Issues** section, we will be introduced to sentiment analysis, what are sentiments, as well as the different approaches to extract sentiment. In the **Application in Python** section, we will create a program to compute sentiment out of online travel reviews. In the **Example from the Real World** section, the reader will be introduced to one of the application of sentiment analysis in the business world: understanding hotel guest experience and satisfaction. Finally, exercises (with their respective solutions) will be made available to the reader.

4.2 Methodological Issues

Sentiment analysis is the study of sentiments and opinions that are contained in text. According to (Hutto and Gilbert, 2014), “sentiment analysis, or opinion mining, is an active area of study in the field of natural language processing that analyzes people’s opinions, sentiments, evaluations, attitudes, and emotions via the computational treatment of subjectivity in text”.

In the context of sentiment analysis, a sentiment is an evaluation towards a word, a sentence, a paragraph etc. that can be categorized as either positive, negative or neutral. To have a better understanding of how a sentiment relates to a sentence, let us consider the following examples and guess the sentiment they convey:

- (1) “I love this hotel, the staff is always helpful!”
- (2) “This restaurant mainly offers African food.”
- (3) “I hate this fitness studio, it is always crowded”

Here, in sentence (1), there is a positive meaning, the author states its subjective opinion, carrying a positive sentiment, towards the hotel and its staff. In example (2), the sentence carries a neutral sentiment; it is just a plain objective fact, without subjectivity and assessment, towards what type of food the restaurant offers. Finally, in the last example (3), it becomes pretty obvious that the sentiment of the text is negative, the fitness studio is described in bad term “always crowded”, and the author states a negative opinion towards the studio “I hate this fitness studio”.

As the reader has probably noticed, it is rather easy for us, humans, to infer the sentiment of a sentence, as we have been trained to understand human languages since the very first day we were born. However, the approach that we used (i.e. sequentially reading sentences and inferring their sentiments) is not really scalable. What if we needed to infer the sentiment of millions of sentences in a short amount of time? We would need lots of competent humans. This seems like a resource intensive task, and there are probably tools to help us: computers! We could define a set of rules that the machine would follow in order to infer sentiment from text. Machines can even compute several tasks concurrently and usually works faster than humans. Therefore, the subsequent question that might arise in your mind is *How to compute the sentiment of textual data?*

There are two main approaches used to compute the sentiment of textual data: the first approach that I would like to introduce you to, is the non-lexical approach, it uses machine learning. This approach uses labeled

training data. Data, mostly words or sentences, even whole paragraphs, are labeled as either having a positive, neutral or negative sentiment. Once the data has been properly labeled, it is then given as input to the machine learning model to train it. Only after it has been trained can the machine learning model be fed the real use-case unlabeled data for which one wishes to know the sentiment orientation.

The second approach to conducting a sentiment analysis, is by means of a so-called sentiment dictionary. In contrast to a normal dictionary, like the Merriam-Webster, a sentiment dictionary is conventionally a small text file database, that does not map a word to its definition or its meaning, but rather, to its sentiment score, which usually lies between -1 and +1, where -1 denotes an extremely negative sentiment while +1 refers to a highly positive sentiment, and 0 a totally neutral statement.

There are numerous non-proprietary dictionaries available on the Web such as SentiWordNet¹, General-Inquirer², ANEW³, VADER⁴. In this chapter, we will solely focus on the VADER dictionary as it is one of the most practical to use. Also, it was purposefully constructed for social media content, which is more often than not, the main source of textual data in a sentiment analysis context.

As explained by Hutto and Gilbert (2014), dictionary-based sentiment analysis also has its advantages over the use of a machine-learning model. In contrast to a machine learning approach, the dictionary approach does not require extensive use of training data (it does however, require, obviously a dictionary); also this approach is not computationally expensive in terms of CPU processing, memory requirements and training/classification time. Machine learning model can demand a lot from a computer, hence the rise of cloud-based machine learning model. Nonetheless, those models are not free and if one does not pay attention carefully, at the end of the day, the bill might be salty. One other advantage of dictionary-based sentiment analysis, is that it is easier to extend: one just needs to add the token and its given sentiment score to the dictionary, and the newly updated dictionary will be immediately ready to be used in production. In contrast, with the machine learning approach, there is a necessity to update the training data by labeling new records, then the updated training data needs once again to be given as input to the model, the model needs to classify the data, and the real use-case data can then be fed to the newly updated model. Labeling data and classifying it requires some time and other resources.

¹<http://sentiwordnet.isti.cnr.it/>

²<https://wjh.harvard.edu/inquirer/>

³<https://csea.phhp.ufl.edu/media.html>

⁴<https://github.com/cjhutto/vaderSentiment>

We will not develop any further on the machine learning approach, but if the reader is curious, Schmunk, Höpken, Fuchs and Lexhagen (2013) provides a good overview of the different ways (including several machine learning methods), to compute sentiment of textual data.

VADER is a sentiment dictionary. VADER stands for Valence Aware Dictionary and sEntiment Reasoner, and is purposefully built for social media user-generated content in the form of text (Facebook posts, Reddit comments, Twitter’s tweets, etc ...). Another key selling point of the VADER dictionary is its ability to detect the valence of a sentiment: the dictionary registers the intensity of the assessed lexical feature. For instance, the word “good” is evaluated to have a compound score of 0.4404, while the words “great” and “best” have respectively a score of 0.6249 and 0.6369. Here, one can clearly observe that, as the intensity of the word increases, its given compound score also further increases too.

In order to build the dictionary, the researchers extracted lexical features that were already present in some of the most popular sentiment dictionaries, such as the General Inquirer or ANEW. Then, they added other lexical feature that are frequently used in the context of social media text, like emoticons, acronyms and slang. Indeed, another selling point of the VADER dictionary is that it is particularly well-suited for social media content. Emoticons convey sentiment and therefore, are relevant enough to also be included in the dictionary.

Then, they used wisdom-of-the-crowd to asses the sentiment of each lexical feature, and only kept those for which there was a large consensus; which resulted in more than 7500 assessed lexical features.

In the next chapter’s section, we will get some practical experience by exploring a dataset of TripAdvisor’s⁵ online travel reviews using the dictionary-based approach to sentiment analysis with the vaderSentiment software.

4.3 Application in Python

For this section, we will use Python3.8⁶, the latest stable release currently available at the time of writing, as well as vaderSentiment⁷, an open-source Python library that provides a simple interface to the VADER sentiment dictionary. Python is a popular general purpose programming language, widely use in web development and scientific computing, including data science. Its syntax is easy to understand and there are plenty of resources freely available

⁵<https://www.tripadvisor.com/>

⁶<https://www.python.org/>

⁷<https://github.com/cjhutto/vaderSentiment>

on the Web. It is highly recommended that the reader has some familiarity with the Python programming language, as well as with the Pandas⁸ package. Those two tools are used a lot in data science, therefore, it is always worth it to invest some time and learn them. For this exercise, there is no need to install either Python or vaderSentiment; a sandboxed environment is provided at labs.play-with-docker.com/⁹. Click on the green **Start** button, then click the **+ ADD NEW INSTANCE** on the left panel, and then run the following command in the terminal (the black screen), omitting the \$ sign (a convention to denote that a command should be executed in a terminal):

```
$ docker pull ryanzidago/sentiment-analysis
```

It will download all the software needed for our hands-on exercise. Once the download is complete, execute the following command:

```
$ docker run --rm -it ryanzidago/sentiment-analysis
```

If all went well, you should see a somewhat similar prompt at the terminal (don't worry if the number after the @ sign is not the same as the one displayed here):

```
fresenius-student@eb62d59a71bd:~/sentiment-analysis$
```

It looks like you are all set, if not, re-read the different steps carefully, you probably might have missed something trivial. Since we will be using the vaderSentiment Python package, the reader is strongly advised to go through the README¹⁰ of the library and familiarize themselves with the software. If at any time, you feel lost, you can always fallback to the code for this actual section on GitHub¹¹.

First, execute the following command to activate the Python shell:

```
$ python3.8
```

We will import the `SentimentIntensityAnalyzer` module to compute the sentiment of a text. Type the following lines within the Python interpreter:

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
```

Now that we have correctly imported the module and saved its name to a variable, we can start using its `polarity_scores()` function to infer the sentiment of a sentence:

⁸<https://pandas.pydata.org/>

⁹<https://labs.play-with-docker.com/>

¹⁰<https://github.com/cjhutto/vaderSentiment>

¹¹<https://gist.github.com/ryanidago/5440bcd66be55ba23ca9b55cf336bab6>

```
sentence = """Best breakfast I have ever had,
but it was way too expensive and the staff was rude and impatient!"""
analyzer.polarity_scores(sentence)
```

Press ENTER and you'll see the following return value of the computation:

```
# => {'neg': 0.276, 'neu': 0.623, 'pos': 0.101, 'compound': -0.6696}
```

vaderSentiment is able to recognize the different sentiments contained in a sentence. **neg** stands for negative, **neu** for neutral and **pos** for positive. Such as in our example, a sentence can carry various sentiment, even opposite ones. That is why there is a **neg** score of 0.276 and a **pos** of 0.101. The first clause ("Best breakfast I have ever had") conveys a positive sentiment while the last one ("but it was way too expensive and the staff was rude and impatient!") a negative one. If, however, one wishes to know the overall sentiment for the whole sentence, then one should look at the **compound_score**. Again, in our example, the sentence is negative, hence the **compound_score** is negative too. According to the library's README, the following scoring can be adopted to interpret the text's **compound_score**:

positive: **compound_score** >= 0.05

neutral: **compound_score** > -0.05 and **compound_score** < 0.05

negative: **compound_score** <= -0.05

vaderSentiment is able to factor-in the intensity of the expressed sentiment based on punctuation. Replace the exclamation mark in the sentence from the previous example with a dot. You will get a milder score:

```
sentence = """Best breakfast I have ever had,
but it was way too expensive and the staff was rude and impatient."""
analyzer.polarity_scores(sentence)
# => {'neg': 0.268, 'neu': 0.63, 'pos': 0.102, 'compound': -0.6369}
```

As you can see, the **compound_score** is equal to -0.6369 while with the exclamation mark, it was equal to -0.6696. Feel free to add more exclamation marks and observe how it influences the actual sentence's sentiment intensity. Then, uppercase all letter of a high-sentiment word, for example, try the following sentence:

```
sentence = """BEST breakfast I have ever had,
but it was way to EXPENSIVE and the staff was RUDE and IMPATIENT!"""
analyzer.polarity_scores(sentence)
# => {'neg': 0.329, 'neu': 0.566, 'pos': 0.105, 'compound': -0.8087}
```

Notice how the sentiment's sentence is of greater intensity compared to our previous examples.

Within the sandboxed environment is a csv file containing real-world reviews from TripAdvisor. Let us explore the dataset, mine some data out of it. First, import the panda library then create a dataframe from the csv file, and finally print out the dataframe's columns to familiarize yourself with the dataset:

```
import pandas
review_dataframe = pandas.read_csv(
    'assets/le_meridien_picadilly.csv',
    parse_dates=['created_date', 'stay_date']
)
print(review_dataframe.keys())
# => Index(['_id', 'created_date', 'published_date', 'title', 'stay_date',
'trip_type', 'value_additional_rating', 'location_additional_rating',
'service_additional_rating', 'rooms_additional_rating',
'cleanliness_additional_rating', 'sleep_quality_additional_rating',
'rating', 'text', 'absolute_url', 'helpful_vote', 'photo_counts'],
dtype='object')
```

To print out one row of the dataset, use the `iloc` function on the dataframe:

```
first_review = review_dataframe.iloc[0]
print(first_review)
```

# => _id	ObjectId(5e983be6d0014436bd2eb16f)
created_date	2020-04-08
published_date	2020-04-08
title	Disappointed
stay_date	2020-03-31
trip_type	NONE
value_additional_rating	NaN
location_additional_rating	NaN
service_additional_rating	4
rooms_additional_rating	2
cleanliness_additional_rating	NaN
sleep_quality_additional_rating	3
rating	3
text	I was taken here for a birthday treat and we w...
absolute_url	https://www.tripadvisor.com/ShowUserReviews-g1...
helpful_vote	NaN
photo_counts	4

The most important observations here are the `text`, which represents the actual text of the review, as well as the `rating`, which represents the actual score given by the reviewer to the hotel. Let's print out the actual text of the review, and try to infer its sentiment with our intuition alone:

```
print(first_review['text'])
# => I was taken here for a birthday treat
and we were very disappointed with the room,
despite being a Junior Suite
it was close to being the most depressing room I've ever stayed in.
It was dark and tired
and we complained and were moved to another room,
only slightly better, very disappointed.
```

The bathroom was tired
and not sure why the bar of soap was replaced every day,
seems a waste of money and bad for the environment.

The air conditioning was not great,
I would hate to be in the room on a warm day,
luckily we were there in March so we managed.

We had access to the Club Lounge which was nice and the staff attentive,
a nice place to relax after a day out in London.

Breakfast was nice,
very extensive but the food started being removed very soon after the end time
so you need to be down for breakfast well before the cut off time.

The gym and pool area was quite good for a hotel.

The younger doorman was very good,
the older one less so, more interested in gossiping with the staff.

Great location but that doesn't compensate for the tired building,
sadly we will not be returning.

Obviously, this review carries a quite negative sentiment! But what does the `vaderSentiment` tells us?

```
analyzer.polarity_scores(first_review['text'])
# => {'neg': 0.144, 'neu': 0.715, 'pos': 0.141, 'compound': 0.3172}
```

Surprisingly, it informed us, that the review is positive (`compound_score >= 0.05`). Maybe the library is not working as expected? As mentioned in `vaderSentiment`'s source code, the software works best at the sentence level. As such, before diving deeply in a data science project it is advised to carefully inspect the source code of the project's external dependencies: there might be valuable information that might save you quite a lot of time and headaches. It is always preferable to find those information while the project is at its infancy stage. In the previous example, we fed the whole paragraph to the `polarity_scores()` function. To resolve this issue, we can chunk the paragraph into sentences:

```
review_text = first_review['text']
sentences = review_text.split(".")
# for every sentence in sentences, compute the polarity_scores()
compound_scores =
    map(
        lambda sentence: analyzer.polarity_scores(sentence)['compound'],
        sentences
    )
```

Now that we have a list of `compound_scores`, we could sum them up all together, or compute the average `compound_score`. Let's sum them up:

```
compound_score = sum(compound_scores)
print(compound_score)
# => -1.2965
```

The computed sentiment is more accurate than before. This was great, but certainly not enough. Let's compute the `compound_scores` for each reviews in the dataset. To do that, we will need to organize our code into small reusable functions. The `compute_compound_score()` function takes one argument, the `text` of the review, that is then splitted into sentences. Finally, the function calculates each `compound_scores` for each `sentences`; sums all `compound_scores` and returns the `compound_score` of the whole review text paragraph:

```
def compute_compound_score(text):
    sentence = text.split(".")
    compound_scores =
    map(
        lambda sentence: analyzer.polarity_scores(sentence)['compound'],
        sentences
    )
```

```

    compound_score = sum(compound_scores)
    return compound_score

```

Now that we have created the `compute_compound_score()` function, we can apply it to the entirety of the dataframe:

```

compound_scores =
[compute_compound_score(text) for text in review_dataframe['text']]
review_dataframe['text_compound_score'] = compound_scores

```

Want to see the result? Try `review_dataframe.iloc[100]` to see the 100th review. At the bottom of the printed message, you will see a new column `text_compound_score` with the newly computed `compound_score` for this review.

This was great. Now, we could categorize each reviews into positive, negative or neutral reviews based on the `compound_score` that we have just obtained. To do that, let us create another function:

```

def categorizes_based_on_text_compound_score(text):
    compound_score = compute_compound_score(text)
    if compound_score <= -0.05:
        print(f"NEGATIVE: {compound_score} -> {text}\n")
        return 'negative'
    elif compound_score > -0.05 and compound_score < 0.05:
        print(f"NEUTRAL: {compound_score} -> {text}\n")
        return 'neutral'
    elif compound_score >= 0.05:
        print(f"POSITIVE: {compound_score} -> {text}\n")
        return 'positive'

```

Here, for a given review text, we calculate the `compound_score` with the help of the `compute_compound_score()` function, then, we categorize the returned result as either positive, negative or neutral, based on the scoring annotations mentioned in vaderSentiment's README. As before, we need to apply the `categorizes_based_on_text_compound_score()` function to the dataframe:

```

categorized_texts =
[categorizes_based_on_text_compound_score(text)
for text in review_dataframe['text']]
review_dataframe['categorized_based_on_text_compound_score'] =
categorized_texts

```

Now, the reader should be able, all by themselves, to inspect the recently updated dataframe to verify if they correctly categorized each reviews.

In the next section, the reader will be introduced to an example of a sentiment analysis project from the real world, where researchers studied the relationship between review rating and sentiment rating.

4.4 Example(s) From the Real World

Geetha, Singha and Sinha (2017) studied the relationship between customer sentiment and online ratings for hotels. They argue that, in order to improve the extremely competitive hotel industry, it is required to improve the understanding of its customers through reviews and ratings that they leave online.

Their main hypothesis is that customer sentiment polarity has a positive effect on customer rating. Geetha et al. (2017) used Simple Random Sampling to select 20 budget hotels and 20 premium hotels, all located in the municipality of Goa, India. To classify reviews as either positive, neutral or negative, they also used a dictionary-based approach. Before feeding the sentences to their own implementation of a sentiment software, they processed the reviews by removing punctuations, numbers, stopwords, white spaces. They also converted all letters to lower case and stemmed every word in reviews.

They found out that the most frequent words in both categories are quit the same (“hotel”, “good”, “room”, “stay”, “staff”, etc.). The results of their study showcase a certain consistency between customer ratings and online travel review sentiments: the better the sentiment, the higher the rating. This also prove us that sentiment analysis is effective, otherwise there would be no correlation between review sentiment and review rating.

Geetha et al. (2017) also discovered that customers from budget hotel were more critical of the hotel where they stayed, with only 55 percent of all reviews having a positive sentiment, while customers from premium hotels were overly satisfied regarding their accommodation, with more than 70 percent positive reviews.

With this valuable information, hotel managers can start to dig deeper to figure out why exactly budget customers aren’t as satisfied as premium customers; it could be that budget hotels in Goa need to reassess their communication strategy by making clear from the very beginning what the customers should expect. Since budget hotels offer by definition, minimal amenities and services, hotel managers could probably increase customer satisfaction and loyalty by improving the way hotel staff interacts with its guests.

4.5 Conclusion

We have learned many things throughout this chapter. We now know that sentiment analysis is about the study of sentiments that are contained within texts. We know that a sentiment is an evaluation of a token, that is categorized as either positive, negative or neutral. We also learned that sentiment analysis is used in a lot of various fields, such as finance, marketing, health-care. We successfully conducted a small sentiment analysis project, by reading data from a dataset, computing the `compound_score` of some reviews, and categorizing reviews as either positive, negative or neutral. We have learned how to use the vaderSentiment Python library to help us in our analysis.

Obviously, this is not all there is to it. There are many more things to learn. If the reader wishes to further their understanding of the topic, they can freely learn from the following resources:

- Web Scraping TripAdvisor, Text Mining and Sentiment Analysis for Hotel Reviews¹². This is an online tutorial for conducting a full sentiment-analysis project, from A to Z, from the blog TowardsDataScience¹³, authored by Susan Li. Completing this more extensive and advanced tutorial would be a great step right after having finished reading and working on the current chapter.
- VADER: A Parsimonious Rule-based Model for Sentiment Analysis for Social Media Text¹⁴. This is the actual research paper of the VADER dictionary. The authors explain why did they built VADER and how did they do it, how did they chose to include some words and some not, etc. This paper is very informative, especially regarding how to build a sentiment dictionary.
- Sentiment Analysis and Subjectivity¹⁵. This is another research paper, authored by Bing Liu, from the University of Illinois at Chicago. In this paper, the author explain very well what is sentiment analysis and what are the challenges pertaining to it.

I hope that the reader enjoyed this chapter as much as I enjoyed writing it. Questions, remarks and improvement ideas can be sent to me by email.

¹²<https://towardsdatascience.com/scraping-tripadvisor-text-mining-and-sentiment-analysis-for-hotel-reviews-cc4e20aef333>

¹³<https://towardsdatascience.com/>

¹⁴<http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf>

¹⁵<https://www.cs.uic.edu/liub/FBS/NLP-handbook-sentiment-analysis.pdf>

Next comes some exercises as well as test questions, to assess your newly acquired knowledge.

4.6 Exercises

After having successfully graduated from the Hochschule Fresenius, you have gotten a position as a junior analyst at a prestigious consulting firm in London. Your manager wants to test you by handing out to you your first big project: helping a struggling hotel to understand their customer: the hotel director, Miss Smith, wishes to know why her customers are satisfied/dissatisfied in order to improve the offered services. She stated that the overall star ratings will not help her better emphasizes with her guests, because those same star ratings provide little context and actionable measures. Instead, she wants to know what is hidden in the comment that her guests publish on online booking platforms.

Within the sandboxed environment is another dataset containing travel reviews from TripAdvisor, for the Premier Inn London Kensington hotel. Your task is to:

1. Find out the percentage (no decimals) of positive and non-negative reviews in the dataset. To proceed, you will need to categorize each reviews as either positive or non-positive reviews, based on the `compound_score` result. This is the stepping-stone for further analysis; this task is crucial.
2. Compute the average `compound_score` (3 decimals places) for the entirety of the reviews. She would like to use it as another metrics to assess the quality of the hotel services.
3. Rank the most satisfied `trip_type` (according to the `compound_score`'s average for each traveler categories). She has decided that she would like to better market the hotel based on data-backed customer segmentation.

Answers:

1. 95% of the hotel guests were satisfied by the hotel itself (consequently, only 5% were not satisfied).
2. The average `compound_score` is equal to 2.077.
3. The most satisfied `trip_type` is the `FRIENDS` trip category, with an average `compound_score` equal to 2.209, followed by the `FAMILY` trips

with 2.163, then the **COUPLES** with 2.061, and **SOLO** trips with 1.944, and finally the **BUSINESS** category with only 1.729.

Here¹⁶ is an example, on how to find out the results.

4.7 Test Questions

1. In the context of sentiment analysis, is *anger* a sentiment or an emotion?
2. Can a sentence carry several opposite sentiments at the same time?
3. What does the `compound_score` returned by the function `polarity_scores()` from `vaderSentiment` represent?
4. If you have a sentence with two clauses, the first one being negative, the second one positive, which clause will impact the most the sentence's sentiment?
5. Does the `vaderSentiment` work best at the paragraph or sentence level? Does it give more accurate score after having computed the sentiment of a paragraph or after having computed the sentiment of a sentence?

Answers:

1. *Anger* is an emotion. The emotion's sentiment would be *negative*.
2. A sentence can carry several opposite sentiments. Let us consider the following example: "The waitress seemed to be under a lot of stress and the pizza arrived quiet late, but it was soooo tasty and worth the wait!". The first clause ("The waitress seemed to be under a lot of stress and the pizza arrived quite late") is a neutral/negative statement, while the second clause ("[...] but it was soooo tasty and worth the wait!") carries a positive sentiment.
3. The `compound_score` represent the overall sentiment of the sentence. If one would need to select only a single sentiment from the sentence, then one would take a look at the sentiment score contained in the `compound_score`.

¹⁶<https://gist.github.com/ryanzidago/ef8093cb553913026b0cd3348cf07c7c>

4. Generally, the last clause of a sentence is the most impactful. You can try to compute the `compound_score` of the example provided in question 2, and then, rearrange the sentence in order to have the last clause at the first position within the sentence, and the first clause at the last (i.e. “The pizza was soooo tasty and worth the wait, but the waitress was seemed to be under a lot of stress and the pizza arrived quite late.”) Observe attentively how much of a difference the position of a clause can make!
5. vaderSentiment works best at the sentence level. It can be used to compute the sentiment of a whole document, or just a word, but it will not be as accurate, as when it is done at the sentence level.

4.8 Glossary

Bigram A pair of tokens conveying meaning together.

Dictionary A database (mostly simply a text file) that maps a token to a score representing the magnitude of its sentiment.

Sentiment analysis Sentiment analysis is the study of sentiments/opinions that are contained in text.

Sentiment In the context of sentiment analysis, a sentiment is an evaluation of a token, that can be either positive or negative. More and more studies also include a neutral sentiment, to have a better granularity in their analysis.

Simple Random Sampling is a statistical method by which one select a smaller sample from a larger one.

Token The smallest entity that can be subject of a sentiment analysis; it can be an emoticon, a word or an abbreviation.

Trigram A set of three tokens conveying meaning together.

User-generated-content User-generated content, also more commonly abbreviated UGC, is the content that is published on the web, by users. It could be Facebook posts, Reddit comment, Instagram pictures, YouTube videos and so on. The vast majority of sentiment analysis deals with textual data.

Valence Valence describes the intensity of a sentiment. Saying that VADER is a valence-aware dictionary means that it is able, not only to determine if a sentiment is either positive, negative or neutral, but also how much positive, negative or neutral the sentiment is, i.e. how intense is the assessed sentiment.

Web 2.0 the Web 2.0 is the new generation of Web, where the users are not a mere consumer of web-content, but can and is encouraged to contribute content to the platforms that they use (Wikipedia, Facebook, Twitter, TripAdvisor, etc ...).

Wisdom of the crowds is the process by which one uses the intelligence of a group of individuals instead of relying on a single expert

Bibliography

- Chung, Siyoung, Mark Chong, Jie Sheng Chua, and Jin Cheon Na**, “Evolution of Corporate Reputation During an Evolving Controversy,” 2019, *23* (1), 52–71.
- Cochrane, John**, “Writing tips for PhD Students,” Technical Report 2005.
- Davcheva, Elena, Martin Adam, and Alexander Benlian**, “User Dynamics in Mental Health Forums – A Sentiment Analysis Perspective,” 2019, p. 15.
- Geetha, M., Pratap Singha, and Sumedha Sinha**, “Relationship Between Customer Sentiment and Online Customer Ratings for Hotels - An Empirical Analysis,” 2017, *61*, 43–54.
- Hutto, C J and Eric Gilbert**, “VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text,” 2014, pp. 216–225.
- Mohan, Saloni, Sahitya Mullapudi, Sudheer Sammeta, Parag Vijayvergia, and David C. Anastasiu**, “Stock Price Prediction Using News Sentiment Analysis,” in “2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)” IEEE 2019, pp. 205–208.
- O’Connor, Peter**, “Managing a Hotel’s Image on TripAdvisor,” 2010, *19* (7), 754–772.
- Provost, Foster and Tom Fawcett**, *Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking*, O’Reilly Media, 2013.
- Schmunk, Sergej, Wolfram Höpken, Matthias Fuchs, and Maria Lexhagen**, “Sentiment Analysis: Extracting Decision-Relevant Knowledge from UGC,” in Zheng Xiang and Iis Tussyadiah, eds., *Information and Communication Technologies in Tourism 2014*, Springer International Publishing, 2013, pp. 253–265.

Vidya, Nur Azizah, Mohamad Ivan Fanany, and Indra Budi,
“Twitter Sentiment to Analyze Net Brand Reputation of Mobile Phone Providers,” 2015, *72*, 519–526.

Xiang, Zheng, Zvi Schwartz, John H. Gerdes, and Muzaffer Uysal,
“What Can Big Data And Text Analytics Tell Us About Hotel Guest Experience And Satisfaction?,” 2015, *44*, 120–130.