Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Project Report

Analysis of news for the prevention of suicides

presented by

Nina Arndt
Studiengang: Software and system development
Matrikelnummer: 6799055

Melanie Budde
Studiengang: Computer science
Matrikelnummer: 6494505

Ariana Sliwa
Studiengang: Human-computer interaction
Matrikelnummer: 6816391

Faculty of Mathematics, Computer Science and Natural Sciences
Department of Computer Science
Research area Scientific Computing

Supervisor: Dr. Julian Kunkel

Hamburg, 2018-03-30

Abstract

This project report describes the analysis of large volumes of news articles for the case of suicide prevention. In detail the handling of the data, the processing and the text mining are described. Furthermore, approaches for the implementation of a sentiment analysis for German language are presented.
The second part of the report describes the procedure to get started with a large amount of medical data and present approaches of processing it.

Contents

1 Introduction

Shorter runs and the battle for audiences influence the media coverage. If news tends to be neutral and without judgment, some daily newspapers/yellow press tend to fall out of the grid and provoke sensitive issues and a style of portraying that seems far from a journalistic commitment. And the myth of the lack of influence of traditional media in times of social networks and co is deceptive - 84% of about 70,000 Germans say in the survey of act [3] that they inform themselves about about current events daily. 42% still use newspapers and to the same extent the Internet as a source of information.
On the other hand, scientists are concerned with the extent to which society is influenced by the reception of certain news.
This project report describes in particular the analysis of large amounts of news articles and the presentation of suicide in them. The aim is to develop a prevention tool that analyzes online published articles on hazardous content and lets the user download a summary of selected news articles and their features.
After explaining the relevance of the connection between suicide and media reception, the authors present the design and technical concept of the application. Subsequently, technical challenges are described and an outlook for further development is given in the conclusion.

2 Media coverage and the relation to suicide

According to the WHO [4], around 800,000 people commit suicide each year. In Germany, there were 10,000 suicides in 2015 [1]. 90% of people who die from suicide suffered from a psychiatric illness, most commonly from depression [7]. The reception of media is a proven risk factor for suicidal people. Scherr [12] provides a comprehensive overview of the current state of research on the relationship between media, depression and suicides.

Werther Effect
Of particular relevance in this context is the so-called "Werther effect". According to this theory, media reports of suicide can trigger additional suicides that would not have occurred without this reporting [9].
Etzersdorfer et al. [6] assume a positive correlation between the extent, duration and prominence of suicide reporting and subsequent imitation acts.

Papageno Effect
In 2010 Niederkrotenthaler et al. [8] found another effect: Articles about depressed people whose coping with the crisis situations described constructively in a media report, have a very positive effect on vulnerable people.

Sentiment
Another point to keep in mind is the influence of general media reception. How do reports of negative events affect depression? Scherr [12] states that this area has been the least explored so far. However, it can be shown that in connection with depression, a more intensive use of media can be observed, as depressive persons increasingly consume media content to operate so-called mood management and to modify their mood positively. In their meta-analysis, Reinemann and Scherr [10] phrase the question of the influence of reporting on wars and catastrophes, but also about unattainable ideals of beauty and wealth. However, this research has not yet developed.

Journalistic guidelines
Because of this well-known influence of the media on people with suicidal thoughts, there are guidelines for reporting suicide in Germany.[2] However, journalists are advised not obliged to abide by them. A positive approach to the reporting of suicide was found in articles from the Süddeutsche Zeitung, which distinguish the danger of imitation and offer help in the form of the number of German crisis telephone. This procedure with the exact same text could also be observed with other media, but not all.

3 Technical Approach to a Suicide Prevention Tool

In this chapter, the authors describe the technical concept and the implementation of the application. Starting with the description of the idea to the technical components such as text mining, sentiment analysis and general execution of the program.

3.1 Design

Assuming that suicide is reported multiple times in local media, a tool for relevant persons, e.g. local psychologists, telephone counseling, etc., could provide information that may be relevant to the treatment of their patients:
How many suicide-related articles appeared on that day? Did the articles offer help for suicidal people? What was the sentimentality of the text - rather positive or rather negative?

The tool has the form of an Excel spreadsheet, which shows the general sentiment of the reporting of the day and shows with the help of a traffic light system at first glance whether there is a threat from the media or not. If a suicide was reported on the day, the articles are listed and the respective sentiment of the article is displayed. A column indicates whether the article offers help to suicidal people, e.g. the request to seek help.

3.2 Technical approach

In the following section, we explain the technical procedure in detail and focus in particular on emerging problems and their (possible) solutions.

3.2.1 Crawler

To get the articles to analyze, a crawler[1] was run from mid-October 2017 to mid-March 2018 that wins news articles from selected rss feeds. For this, the rss feeds are checked for new articles at a fixed interval. In this project, it was decided to include the ten most widely read news sites in Germany and the ten most widely read news sites in Hamburg, the complete list can be found in the appendix (5.3.3).

[1] crawly by Max Luebbering, https://github.com/le1nux/crawly.git

There were some challenges working with the crawler. For one thing, the server has been down for a few days, which means that the data is incomplete. However, this hardly plays a role in the question of this work, so that the results are meaningful despite the lack of data. On the other hand, paywalls prevented the article texts from being saved on some of the selected pages. A solution to this problem has not been achieved in the context of this work, so that for some media, the article texts are incomplete. This distorts the results of the sentiment analysis, which has to be considered in the evaluation. Nevertheless, in order to be able to perform the keyword search for all articles, the article description was included, which is available for all articles.

3.2.2 Preprocessing

Since the crawler saves the complete html text of the respective article page, it must be cleaned up in order to be able to continue working as a running text. For this purpose, a filter was written for each medium considered in this project, which finds the entypoint as the html-tag of the article text. From this point the text is saved for further processing. Furthermore, html-tags are defined, which will be deleted in the remaining text.
Difficulties occurred because the downloads did not contain all the html text, but only JSON data. Here the filters can not find an entypoint. The authors chose to adopt the raw text as a whole, since the programming terms have no impact on sentiment analysis and keyword search.

```
filters = {
    "not_impl_filter": {},
    "newsfeed.zeit.de": {
        "standard_filter": {
            "entrypoints": [{"tag": "article"}],
            "delete": [
                {"tag": "script"},
                {"tag": "style"},
                {"tag": "figure"},
                {"tag": "input"},
                {"tag": "div", "attrs": [
                    {"attr": "class", "values": ["metadata", "article-heading",
                        ↪ "article-footer", "sharing-menu"]}
                ]}
            ]
        }
},
```

Listing 3.1: Filter for www.newsfeed.zeit.de

The crawled articles were stored on the cluster throughout the period under consideration and adjusted there in mid-March. Prior to this point, only individual files were downloaded and filtered for testing purposes. The filtered files were then stored on local computers and processed there.

3.2.3 Text mining

Text mining is about the automatic extraction of previously unknown information from unstructured text sources. The goal is to put information in context and draw conclusions

from it. This task is commonplace and manageable for humans, but more difficult to implement through algorithms and programs. The text documents to be examined are available in different forms, so that a pre-structuring of the data is required.[13] For this purpose, various packages and libraries are provided in Python, which are suitable for processing natural language, e.g. the natural language processing toolkit (nltk).

In this project, it made sense to divide the body text into individual words, e.g. to perform a keyword search, therefore, the article texts were subdivided into tokens using the nltk word_tokenize function. Furthermore, before analyzing the text, words that are important to the grammar but that are irrelevant to the meaning of the text (e.g., conjunctions) must be deleted. Again, nltk provides a function that also recognizes German stopwords.

Keyword search to detect articles that address suicide

To detect the risk of a possible Werther effect, the crawled articles are searched in this project with a simple keyword search for the german word "Suizid" or synonyms for it.

```
1 def findKeywordSuicide(text):
2     keywords = ['selbstmord', 'suizid', 'leben genommen', 'sich umgebracht',
          ↪ 'freitod', 'selbsttoetung', 'lebensmuede', 'leben beenden',
          ↪ 'selbstentleibung', 'selbstvernichtung']
3     found = 0
4     lowertext = text.lower()
5     for k in keywords:
6         if k in lowertext:
7             found = 1
8     return found
```

Listing 3.2: Keyword search

If articles are found that contain at least one of these words, these articles will be searched for help and support for those affected. In this case, the phone number of the "Telefon Seelsorge" (crisis telephone) is searched because it can be assumed that help offers contain this number (see chapter 2).

The problem with this approach is that it does not check what kind of text it is. Accordingly, a police report on a local suicide is not distinguished from a fictional text, a literary or film criticism in which a suicide is mentioned.

Sentiment Analysis

Sentiment analyses are based on the principle that text is analyzed on the basis of certain training data. For example, with a certain probability, the text is more likely to be positive or rather negative. These training data can be either very general or very specific to the text to be analyzed.
The authors have found that for news articles in Germany no training data exist. One possible reason for this might be that journalists are generally required to write their report neutrally and without judging as much as possible.

Nevertheless, the authors develop two different approaches to analyze the sentiment of the gained data.
The first approach uses a lexicon as training data, in which positive and negative words have been classified. The second one learns by texts from the Internet. Both approaches are explained below.

Sentiment Analysis with Dictionary Method

In this approach of sentiment analysis, the text to be classified is decomposed into individual tokens and matched with a dictionary in which the words are labeled and given a particular weighting with which they refer to the associated label. In the present implementation, the German-language dictionary "SentiWS" of the University of Leipzig [11] is used, which contains 1,650 positive and 1,818 negative words. The weighting of the words is between -1 (negative) and 1 (positive). In the present implementation, after being tokenized and cleaned up by stopwords, the texts to be classified are matched with the positive and negative dictionaries, and the weighting of the found words is added to or subtracted from the sentiment value. Finally, the sentiment value is being normalized to the text length.

```
1  def sentiment_extraction(elem):
2      text = elem[5]
3      sentiment = 0
4      tokens = getFilteredTokens(text)
5      for t in tokens:
6          for elem in negative_words:
7              if t in elem:
8                  sentiment += float(elem[1])
9          for elem2 in positive_words:
10             if t in elem2:
11                 sentiment += float(elem2[1])
12     if (len(tokens)>0):
13         sentiment = normalize((sentiment/len(tokens))*100)
14     return sentiment
```

Listing 3.3: Sentiment analysis with dictionary

Sentiment Analysis with Naive Bayes Classifier

The Bayes Classifier is a classifier based on a mathematical theorem of probability theory. Different classes are defined for the assignment, for example different subject fields or attributes like "positive" and "negative". A function calculates the probabilities of an object's properties to belong to a particular class. When classifying the entire object, the probability of the common occurrence of certain properties is included in the calculation. In an example with three possible classes arranged in a triangle for a visual representation, the classifier would start near the class that has the highest frequency. Then, for each property of the object to be classified, a vector is drawn whose direction depends on the class to which this property points. The length of the vector is determined by the weighting of the probability of this assignment. The result is the class closest to the end of the classification. [5]

In the present implementation of the Bayes Classifier, a sentiment analysis of German-language news articles is performed. As training data serve different texts on happy or sad topics, which are classified as "positive" or "negative". The selection of the training data has been made according to subjective feelings and does not follow any defined criteria.

```
def classification(elem):
    neg_texts = getWordList(neg_files, "negative")
    pos_texts = getWordList(pos_files, "positive")
    train_set = neg_texts + pos_texts
    classifier = NaiveBayesClassifier.train(train_set)

    if isinstance(elem[5], str):
        words = getDictionary(elem[5])
        sent_value = 0
        result = classifier.classify(words)
    if (result == "positive"):
        sent_value = 1
    return sent_value * 100
```

Listing 3.4: Sentiment analysis with Naive Bayes Classifier

For the current state of the project, the training texts are each 10 texts of Wikipedia entries which occur to the authors as positive or negative perceived topics. For English-speaking countries, there are databases with training data on different types of text, but this is unfortunately not yet available for German-language texts, which makes the generation of training data more complex. Presumably, better prepared training data which are classified according to comprehensible criteria, would significantly improve the results.

Topic classification with Naive Bayes Classifier

Due to the difficulty of dividing news articles that claim to be neutral into "positive" and "negative" and the fact that the influence of depressive people due to the sentiment of a text is controversial, the idea came up while working on this project, that any other text classification parameters might be more useful.

With the Bayes Classifier it is possible to determine the topic of an article using a training dataset. As part of this project, the three topics "war", "diseases" and "sport" were trained with five articles each. A larger scope of topics and training data could not be implemented due to time constraints, but would be mandatory for a meaningful evaluation. Accordingly, the present implementation can only be seen as an idea for possible further developments. Furthermore, it remains to be examined whether a classification of the topics from a therapeutic point of view would be useful and could be used in the context of a therapy. A starting point could be that therapists have the opportunity to specifically target patients who are triggered by certain topics, if they are discussed more often than average.

3.2.4 Implementation and deployment

To visualize the data, two different approaches were used in this project. First, an Excel sheet can be created per day. For this the keyword search and both approaches of the sentiment analysis are carried out in all articles of the csv file. The average sentiment value of the day is added to the Excel sheet, as well as the articles are listed, which contain keywords that indicate a suicide. Furthermore, a field red, yellow or green is marked and should represent as a kind of warning system. If a suicide was mentioned the field is filled in red (see : 3.1). In the case that no suicide was mentioned and the sentiment value is positive the field is filled in green (3.2). This type of visualization is intended as an offer for therapists working with suicidal and depressive people. The analysis could be done on a daily basis at the cluster of DKRZ and send therapists the Excel sheet by e-mail, so that the following day they can talk to their clients about possible suicides or particularly negative-tuned articles.

	A	B	C	D	E	F	G
1	**Suicide Prevention information based on media published on 2018-3-27-16-44 File: ./Website_Downloads_2017-11-25.Csv-Cleaned.Csv**						
2	Sentiment with dict	49,845116605					
3	Sentiment with BayesClassifier	0					
4							
5	At least one article was found that contains the word suicide:						
6	**Medium**	**Title**	**Link**	**Sent1**	**Sent2**	**Help offered**	
7	www.tagesschau.de	Gewalt in Pakistan: 200 Verletzte bei Protestcamp-Räumung | tagesschau.de	http://www.tagesschau.de/au	48,5559	negative	0	
8	newsfeed.zeit.de	Rapper Lil Peep: Drogentod auf Instagram | ZEIT ONLINE	http://www.zeit.de/kultur/mus	50,0043	negative	0	
9	www.bild.de	Grusel-Fund in Tostedt - Mann hängt tot im Strommast | bild.de	http://www.bild.de/regional/	49,3571	negative	0	
10							
11							

Figure 3.1: Excel sheet for 25th November 2017

	A	B	C	D	E	F	G	H	I	J
1	**Suicide Prevention information based on media published on 2018-3-16-14-40**									
2	Sentiment with dict	66,3054								
3	Sentiment with BayesClassifier	0								
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										

Figure 3.2: Excel sheet for 12th December 2017

Second, some of the calculated values were visualized in diagrams. For example, the two sentiment values can be displayed as a bar chart per medium. Furthermore, the average sentiment value using the Dictionary method per medium was displayed as a line chart over the week. As can be seen in the following graphics 3.3, 3.4, 3.5, it is noticeable that the values hardly differ between the different media and that there are no major fluctuations over time. This is probably partly due to the fact that the articles are written relatively neutrally, as it corresponds to the journalistic standard. Partly better training data would probably also contribute to a better result, that is more differentiated.

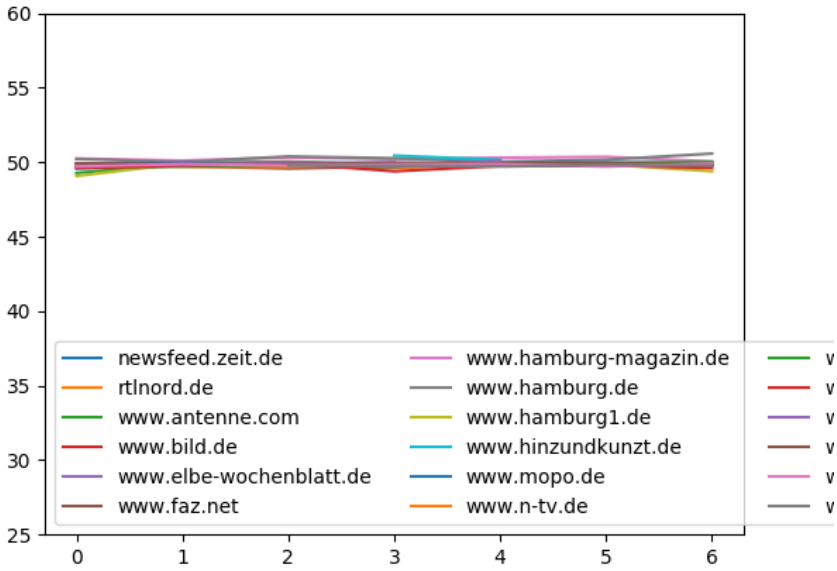

Figure 3.3: Mean sentiment value for 19th to 25th of February 2018

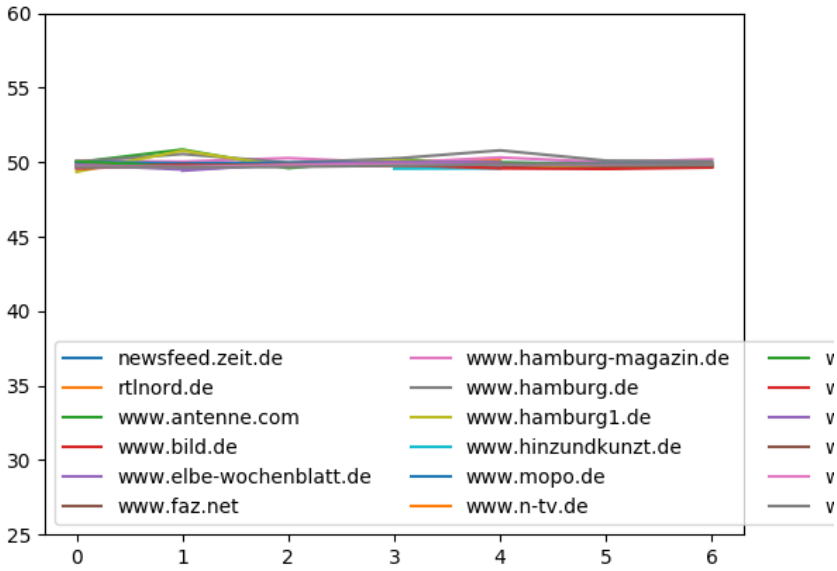

Figure 3.4: Mean sentiment value for 13th to 19th of November 2017

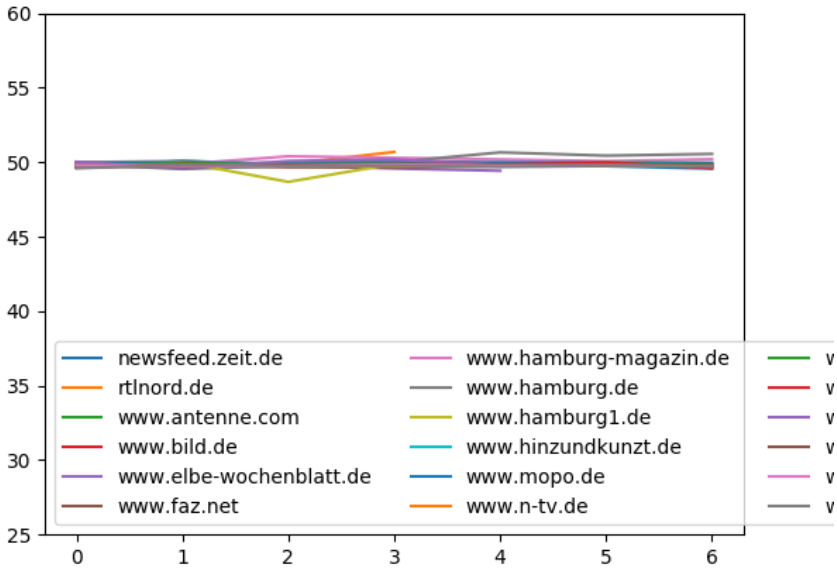

Figure 3.5: Mean sentiment value for 20th to 26th of November 2017

3.2.5 Performance

The following section lists some of the runtimes of the code created in the project to assess its performance:

- One csv file per day (at 110 days) with all articles of the day was filtered on the cluster. Here, 15 cores were used. Running time: 45 minutes.
- The function create_Excel was executed on a local computer with intel core i7 and 16 GB of RAM. Running time per csv file (about 25 MB): approx. 15 minutes.
- The average sentiment values with dictionary per day over one week were also calculated on a local computer (intel core i7, 16 GB of RAM) and plotted in a line chart. Running time for seven csv files: about 45 minutes.

3.3 Technical challenges

The work on this project posed some challenges to the authors: Two out of three people had previously not worked with the programming language Python and had to get used to it. Furthermore, the handling of the cluster had to be learned. After a substantive introduction to the topics of text mining and sentiment analysis, the problem arose that many of the libraries and packages available in this area are designed for the English-speaking world, so for English texts there are provided Databases with training data that are not available for German texts. In the group, the impression arose that basically there are few examples of a sentiment analysis for the German-speaking world. As a result, it took a little longer to get into this topic, but it was also possible to gain a good insight and basic understanding of this area of work.

When working with the crawler, this project benefited greatly from the fact that it was possible to draw on the preparatory work of another student. So crawlers and filters had to be modified and adapted to the required newsfeeds for this project. Nevertheless, some problems arose in this area: For example, the texts were not transferred correctly on some pages, as some media apparently prevented this by a paywall. Furthermore, the filters had to be adjusted individually for each medium several times, which was time consuming.

4 Conclusion

As a conclusion from the project can be found that the keyword search works well and finds all articles with the specified keywords. However, there are often articles that do not report a real suicide, but use the word for example as a metaphor. In order to achieve more differentiated results, a possibility would have to be developed to further analyze the texts found and assign them more precisely to a genre. One possible approach for this is, for example, the topic classification with the Naive Bayes Classifier, which has been implemented in the present work.
Sentiment analysis using the dictionary method works well. However, the classification of news articles into positively and negatively tuned texts makes only limited sense in this case. This also explains the barely changing values.
The sentiment analysis with the Naive Bayes Classifier has not a high performance and gives only little satisfactory results. However, this is probably largely due to the lack of quality of the training data. The authors came to the conclusion that differentiated selected data would lead to significantly better results and that it would be worthwhile to continue working in this direction.
Furthermore, it could be exciting to analyze the individual articles in more detail. For example, it might be enlightening to have the most common words printed in articles. This could be a clue to identifying which words most often lead to a positive or negative sentiment value. It would also be interesting to use an advanced Bayes Classifier for topic classification to identify the most important topics of the day.
As mentioned in Chapter 2, the state of research on the impact of media content on people with depression is still relatively unexplored. Future research, based on the sentiment analysis related to the topics of the reports, could address a link between these and other statistics in the field of suicide and depression.

5 UKE-Data

5.1 Introduction

In the beginning of this project, one of the main approaches was to find correlations between suicides and news articles about this topic. Therefore the Psychology-Department of UKE wanted to reveal a data-set with suicide information from the pathology. That did not work out because the contact was not available anymore, so we got another set of data. The following chapter is an additional excurse to the main topic, containing data exploration from a different data-set and the implementation of a socket connection for data exchange.

5.2 Data exploration

The Department of Neurophysiology at UKE is working on an experiment to gain knowledge about the correlations between interaction, empathy and social health. Therefore they developed a computer game where two subjects have to coordinate one ball to collect coins and avoid obstacles together. They provided two matlab files containing meta information about the attended subjects and parameter-value pairs for each trial of the game. In this section Python was used and especially the libraries:

- numpy
- scipy.io
- matplotlib

to take a closer look at the data and understanding the game which was not seen or played by the authors yet. All that was known, is that there are obstacles an coins which have to be collected by two players controlling one ball. Further knowledge about the game was gained by the following two chapters.

5.2.1 Preprocessing

First of all scipy.io is used to store the mat file into a dictionary with variable names as keys, and loaded matrices as values, so the data is accessible via key or index.

```
1 game = scipy.io.loadmat('Data_BallGame_6pairs.mat')
2 player = scipy.io.loadmat('Data/meta_info_6pairs.mat')
```

This also allows to get meta-information about the data, for instance the dimension of the matrices or data type.
game = (6, 3)
player = (2, 2)

For further examination,

- **the properties of *game* were stored in a file → properties.py**
  It is easier to read and extract data from *game* if the indices from the dict are connected to their value-keys.
- **the subject properties of *player* were stored in a class → subject.py**
  It creates an instance of subject which has access to it's properties via a function and returns the information needed. This way it is possible to easily extract data from the set.

This is the foundation for a more complex analysis.

5.2.2 Analysis

Within the structure that was created for analysis, it is possible to ask more complex questions in *extract.py*. For example: How is the age-distribution among the subjects in *player*?

```
1  def age_distribution(card_sub, teen , twen, old):
2    x = teen
3    y = twen
4    z = old
5    card_game = 0
6    while card_game < len(data):
7      age = sb.Subject(card_sub , card_game).age()
8      if age <= 20:
9        x = x + 1
10     elif 21 <= age <= 30:
11       y = y + 1
12     else:
13       z = z + 1
```

```
14      card_game = card_game + 1
15    card_sub = card_sub + 1
16    if card_sub < 2:
17      age_distribution(card_sub, x, y, z)
18    else:
19      print("%d are younger that 20" % x)
20      print("%d are between 20 and 30" % y)
21      print("%d is older than 30 \n" % z)
```

This example of a simple query can be exposed to various complexity, like:
How is the age-distribution among the subjects according to their age in *player*?

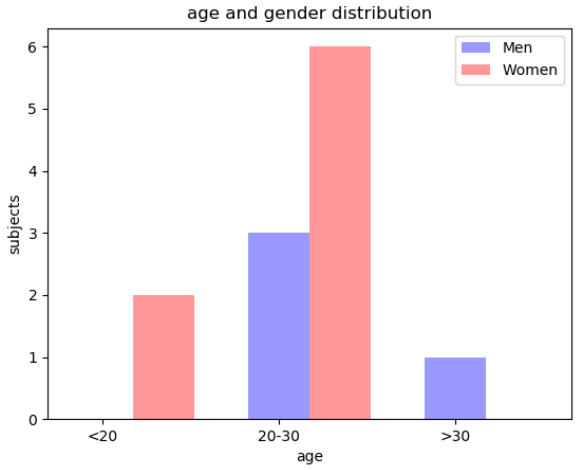

Figure 5.1: Metainfo about subjects

To get a better understanding of how the game actually works, the x and y coordinates of the ball and location of the obstacles and coins for each frame in one trial were plotted. Matplotlip was used not only for plotting, but for a little animation of one game in *anime.py*:

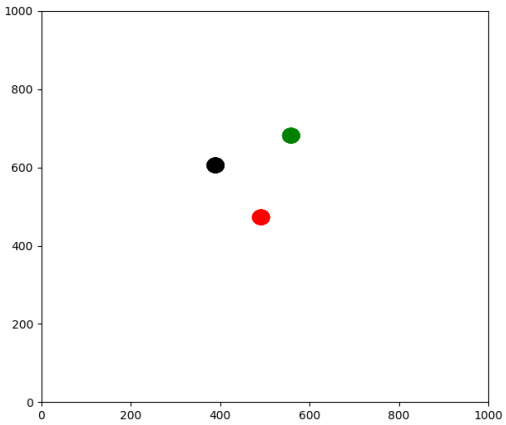

Figure 5.2: Animation of game-play

Every time the ball hits an obstacle it turns red and whenever it is collecting a coin, it turns green. The following knowledge was gained by the animation:

- There are 9 obstacles and
- 4 coins to collect.
- These items are placed randomly in every trial
- The obstacles do not have a physical body which stops the ball, it only slows down.

5.3 Socket implementation

The ball-game from previous section is implemented in LabView, a system-design platform and development environment for visual programming. In future for an upcoming bachelor-thesis there will be an artificial intelligence implemented, that is capable to play the game against one subject and behave in certain ways, depending on the personality coefficients and correlations gained from subject's meta-data. Python will be used for the implementation of the AI. In order to exchange and typecast the data-type 'cluster' from Labview to Python, a socket connection is needed that is capable of taking JSON via TCP to process the data for purposes of communication in Python.

5.3.1 Set-Up

To combine Linux advantages in software-development, like paket-manager and development-tools, with Windows based LabView, a second environment and connection between them is needed. VirtualBox was used to install Windows 10 on a virtual machine, embedded in the Linux host-system. The installation of LabView and all the dependencies needed in the project like matlab and the used packages and engines, took nearly three days and the computer had to be in the subnet of the Department for Neurophysiology at UKE. For future collaboration on the project, the network will be configured, so it can be possible to access the data via UHH and VPN. To establish a connection between the host and the virtual machine, a host-only adapter was configured.

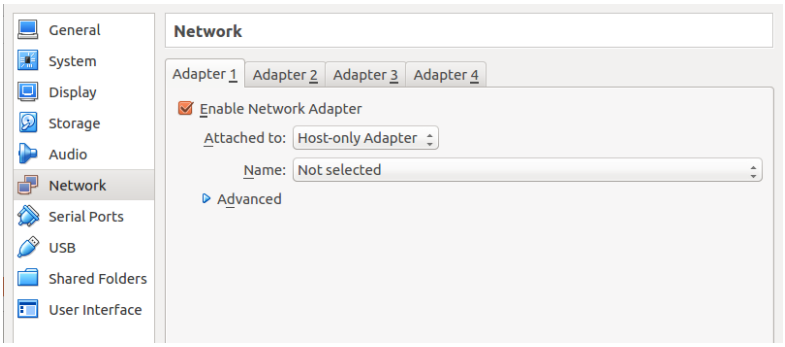

Figure 5.3: Interface in virtual box

- **NAT** allows the vm to connect to the host's network but also establishes a router/firewall which protects the vm from other devices, so this is not suitable for purposes of a client-server model.
- **Bridge** fits for applications which have to be accessible within a network for many clients, which is not the case here.

- **Host-only** creates on the vm and hostsystem a virtual network-interface which manages the communication via IP and suites our needs for data-exange.

These steps have been necessary to connect both systems in general. This is the setup to implement a proper socket connection.

5.3.2 Data-Exchange

The communication is realized as client-server model where the Python application serves as client and LabView as the server. Due to easy typecasting the format of data that will be exchanged is JSON and will be transmitted using TCP sockets.
This is the code for the python-socket:

```
1  import socket
2  import json
3
4  TCP_IP = '172.18.101.69'
5  TCP_PORT = 1337
6  BUFFER_SIZE = 1
7  MESSAGE = "Hello, World!"
8  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9  s.connect((TCP_IP, TCP_PORT))
10
11 fr_rec = False
12 byteA = bytearray()
13 while not fr_rec:
14     data = s.recv(BUFFER_SIZE)
15     if data[0] == 10:
16         fr_rec = True
17     else:
18         byteA.append(data[0])
19
20 a = json.loads(byteA)
21 s.send(MESSAGE)
22 s.close()
```

.
Normally a buffer-size is set and if the received data is smaller than the buffer-size it should work as well, but there are time-outs. The workaround is to set the buffer-size to 1, set a stop flag at the end of each frame and loop over each input bit until the If the stop-flag (data[0]==10) at the end of each package arrives. It is important to have buffer-size of 1 because otherwise the stop-flag could not be recognized. If the buffer-size is 15, it means that 15 chars will be transmitted, but our break condition only considers the first bits for checking, this could cause timeouts as well. At this state a fluid connection between the two systems is established.

5.3.3 Further thoughts

It is not clear yet, if this simple exchange is enough. Right now the socket connection is able to receive and send data. For improved interaction, another loop could be wrapped around the exchange (line 11-21), switching between sending and receiving data. That could be one way to analyze incoming values and calculate the AI's response. The reason this is not done, is because it is not exactly clear yet, how the AI will communicate and if it has to be re implemented with Burkley-Sockets like ZeroMq for being able to have several Ports to address different services of the application.

Bibliography

[1] Statistisches Bundesamt. Todesursachen. URL https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/Gesundheit/Todesursachen/Todesursachen.html. Accessed: 14.03.2018.

[2] Leitlinien zur Suizidberichterstattung. URL http://frans-hilft.de/presse.

[3] ACTA - Berichtsband, Nov 2016. URL www.acta-online.de. Accessed: 14.03.2018.

[4] WHO. Suicide Fact sheet, 2018. URL http://www.who.int/mediacentre/factsheets/fs398/en/. Accessed: 14.03.2018.

[5] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[6] Elmar Etzersdorfer and Martin Voracek and Gernot Sonneck. *A Dose-Response Relationship Between Imitational Suicides and Newspaper Distribution*, volume 8. Routledge, 2004. doi: 10.1080/13811110490270985. URL https://doi.org/10.1080/13811110490270985. PMID: 16006399.

[7] José Manoel Bertolote and Alexandra Fleischmann and Diego De Leo and Danuta Wasserman. Psychiatric Diagnoses and Suicide: Revisiting the Evidence. *Crisis*, 25(4):147–155, 2004. doi: 10.1027/0227-5910.25.4.147. URL https://doi.org/10.1027/0227-5910.25.4.147. PMID: 15580849.

[8] Thomas Niederkrotenthaler, Martin Voracek, Arno Herberth, Benedikt Till, Markus Strauss, Elmar Etzersdorfer, Brigitte Eisenwort, and Gernot Sonneck. Role of media reports in completed and prevented suicide: Werther v. Papageno effects. 197(03): 234–243, sep 2010. doi: 10.1192/bjp.bp.109.074633. URL https://doi.org/10.1192/bjp.bp.109.074633.

[9] David P. Phillips. The Influence of Suggestion on Suicide: Substantive and Theoretical Implications of the Werther Effect. 39:340, 06 1974.

[10] Carsten Reinemann and Sebastian Scherr. Der Werther-Defekt Plädoyer für einen neuen Blick auf den Zusammenhang von suizidalem Verhalten und Medien. *Publizistik*, 56(1):89–94, 2011. doi: 10.1007/s11616-010-0109-y.

[11] R. Remus, U. Quasthoff, and G. Heyer. Sentiws – a publicly available german-language resource for sentiment analysis. In *Proceedings of the 7th International Language Resources and Evaluation (LREC'10)*, 2010.

[12] Sebastian Scherr. *Depression – Medien – Suizid - Zur empirischen Relevanz von Depressionen und Medien für die Suizidalität*. Springer-Verlag, Berlin Heidelberg New York, 2015. ISBN 978-3-658-11162-5.

[13] Sholom M Weiss, Nitin Indurkhya, and Tong Zhang. *Fundamentals of predictive text mining*. Springer, 2015.

Appendices

Work distribution within the group

Crawler & filter

code:	Max Lübbering
adjustments:	Ariana Sliwa, Melanie Budde, Nina Arndt
report:	Ariana Sliwa, Nina Arndt

Preprocessing

code:	Ariana Sliwa, Nina Arndt
report:	Ariana Sliwa, Nina Arndt

Textmining

code:	Ariana Sliwa, Nina Arndt
report:	Ariana Sliwa, Nina Arndt

Impl. and deployment

code:	Ariana Sliwa, Nina Arndt
report:	Ariana Sliwa, Nina Arndt

UKE-Data

code:	Melanie Budde
report:	Melanie Budde

Used software, languages and libraries:

Languages

Python

Libraries

beautifulSoup
nltk
pandas
matplotlib
scipy.io
socket
json

Software

code: Pycharm, JupyterLab, VirtualBox
report: LaTeX, Overleaf

Media List used to install the crawler

National Media:
10 most visited news sites BRD Sep 2017:
(Source: https://de.statista.com/statistik/daten/studie/165258/umfrage/reichweite-der-meistbesuchten-nachrichtenwebsites/)

http://www.focus.de/
http://rss.focus.de/fol/XML/rss_folnews.xml

http://www.bild.de
http://www.bild.de/rssfeeds/rss3-20745882,feed=alles.bild.html

http://www.spiegel.de/
http://www.spiegel.de/schlagzeilen/index.rss

http://www.welt.de/
https://www.welt.de/feeds/latest.rss

http://www.zeit.de/
http://newsfeed.zeit.de/all

http://www.sueddeutsche.de/
http://www.sueddeutsche.de/updates-rss

http://www.n-tv.de/
http://www.n-tv.de/rss

http://www.stern.de/
https://www.stern.de/feed/standard/alle-nachrichten/

http://www.faz.net/aktuell/
http://www.faz.net/rss/aktuell/

http://www.rp-online.de/
http://www.rp-online.de/feed.rss

Regional Media:
(Source: https://www.hk24.de/blob/hhihk24/servicemarken/presse/downloads/1141414/e144b33483
2016_2017-data.pdf -> Some media no longer exist, have been summarized as ePaper without Rss or do not offer a Rss feed at all)

https://www.abendblatt.de/
https://www.abendblatt.de/hamburg/?service=Rss

http://www.mopo.de/
https://www.mopo.de/feed/index.rss

http://www.zeit.de/hamburg/index
http://www.bild.de/regional/hamburg/hamburg-regional/home-16344102.bild.html

Anzeigen- und Wochenblätter

www.elbe-wochenblatt.de
http://www.elbe-wochenblatt.de/feed/action/mode/realm/ID/0/

www.nie-wo.de
https://www.niendorfer-wochenblatt.de/feed/

www.hamburger-wochenblatt.de
http://www.hamburger-wochenblatt.de/feed/action/mode/realm/ID/0/

www.ndr.de
http://www.ndr.de/homepage985-rss2.xml

www.radiohamburg.de
http://www.radiohamburg.de/rss/feed/meldungen

http://rtlnord.de/hamburg-schleswig-holstein.html
http://rtlnord.de/share/rss.xml

www.antenne.com
https://www.antenne.com/rss.xml

www.hamburg1.de
http://www.hamburg1.de/feed.rss

www.hamburgschnackt.de
https://hamburgschnackt.de/feed/

www.hamburg.de
http://www.hamburg.de/,rss

www.hamburg-magazin.de
https://www.hamburg-magazin.de/rssfeed

www.hinzundkunzt.de
https://www.hinzundkunzt.de/feed/

http://www.taz-hamburg.de/
http://www.taz-hamburg.de/!p4608;rss/

http://www.tagesschau.de
http://www.tagesschau.de/xml/rss2/

Polizeimeldungen
http://www.presseportal.de/blaulicht/l/hamburg
http://www.presseportal.de/rss/polizei/laender/5.rss2