CITYSENT

Sentiment analysis of the Netherlands and Flanders

Daan Goslinga, Raoul Kalisvaart, Adriaan Pardoel, Cathrine Paulsen & Sarah de Wolf

24 June 2020

CITYSENT

Sentiment analysis of the Netherlands and Flanders

by
Daan Goslinga, Raoul Kalisvaart, Adriaan Pardoel, Cathrine Paulsen & Sarah de Wolf

to complete the Bachelor End Project
at the Delft University of Technology,
to be publicly presented on Wednesday July 1, 2020.

Project Coach:	G. Migut,	TU Delft EEMCS
Project Clients:	Dr. E. J. Meijers,	TU Delft Urbanism
	Dr. C. E. L. Newton,	TU Delft Urbanism

Project duration: April 20, 2020 – 26 June, 2020

Preface

This report is part of the Bachelor End Project (BEP), in which a group of students does a project for a client. In our case, we have been given the assignment to create a tool that can gather data from social media and display the sentiment associated with this data. Our clients, Evert Meijers and Caroline Newton, want to see sentiment in different places, times, ages or genders and compare them to each other.

We would like to express our gratitude to Gosia Migut for guiding us throughout this project and the flexibility she has given us during the Corona pandemic. Furthermore, we would like to thank Evert Meijers and Caroline Newton for their cooperation with the team and their clear communication of their vision for the project. Lastly, we want to thank Alessandro Bozzon for giving us relevant information about data collection and sentiment analysis.

Delft, June 2020

Summary

The department of Urbanism at the TU Delft, our clients, research the sentiment in different places, times, ages or genders and compare them to each other. This report describes the purpose, design, implementation and accuracy of a web tool created to get insights into the sentiment people have, deducted from social media. The aim of our project was to make research easy and extracting innovative insights from social media.

In our tool, we analysed tweets and their location to collect information about the sentiment different people have towards places. The implementation of our tool consisted of five main components: (1) Twitter-Kafka, processing the tweets from the tweet data stream to our database, (2) face recognition, used for determining whether a tweet comes from a person instead of a company or organisation and for age and gender inference, (3) sentiment analysis, using machine learning to determine whether a tweet is neutral, negative or positive, (4) REST API, for the connection between the front-end and the back-end and (5) the user interface, in the form of an interactive dashboard.

At the beginning of the project, we set up a pipeline that checks the code on multiple things. The testing of the back-end is based on a Python unit test suit. For the build to succeed, all tests must pass and the total branch coverage must be at least 80%. We used Flake8 and ESlint in our build to ensure code quality at all times.

All of the above-mentioned components are up and running. The clients are now able to research the sentiments of people towards places.

Contents

1 Introduction

With the advent of social media, big data has become an increasingly valuable commodity. Data from social media can provide insight into people and how they react. These insights are useful for conducting sentiment analysis, e.g. to investigate the trends of people showing negative, neutral and positive sentiment towards places. Knowing how people feel about certain places and what influences their sentiment, is especially valuable for urban planners and researchers that aim to improve city design as it can help motivate their decisions based on data.

The main goal of the project's clients is to do research on the sentiment of different places and determine which factors influence said sentiment. To facilitate the analysis of sentiment data, the clients want an easily expandable tool that can gather sentiment data and display it in a dashboard. To realise this goal, the web application CITYSENT was created.

CITYSENT is a research tool that combines sentiment analysis and geographical data extracted from the Twitter data stream within the Netherlands and Flanders. CITYSENT aims to make sentiment analysis related to geographical places easier and more accessible for urban planning and research. It does so by automating data collection, performing sentiment analysis and displaying the results in a simple and intuitive way to the end-user. CITYSENT's purpose is not to find correlations between certain factors and sentiment, but to provide the clients with the necessary data so that they can deduce these correlations themselves. Instead of researchers having to manually process the data, CITYSENT will process and analyse the incoming data stream from Twitter in real-time by using stream processing. Through machine learning, the sentiment of the tweet is returned together with the age and the gender of the tweet author. The end-user can use CITYSENT to quickly get an overview of the collected sentiment data from various places, but also download the data for themselves to do further research if necessary. Filter options are also available to facilitate research on certain age groups, genders, specific places and time periods.

This report aims to elaborate on which design decisions were made in the making of CITYSENT to solve the clients' problem. Furthermore, it explains how CITYSENT is implemented and whether these implementations solved the stated problem. After this, the accuracy of the data provided by CITYSENT is discussed. Lastly, a process evaluation of the project and future recommendations will be given. Ethical implications of the project are also discussed.

2 Design

2.1. Overview

The CITYSENT system comprises three main components: the web application, the stream processor and the database that connects the two, see figure 2.1.

Twitter-Kafka, the stream processing component, is responsible for the data collection pipeline. Using the Twitter Stream API and Apache Kafka, Twitter-Kafka turns the raw stream of tweets provided by Twitter into a stream of processed tweets that are stored in a MySQL database.

The web application is split up into two individual components: the frontend and backend. The frontend is built upon a React framework and provides an interactive dashboard to the end-user. Through the dashboard, the user can see and interact with various aspects of the processed Twitter data, such as tweet sentiment and age and gender distribution of the tweet authors. Whenever the user requests to see certain data, the frontend forwards this request to the backend. The backend is responsible for all data requests; it uses a RESTful API to query the database to fetch the processed tweet data. The backend serves the frontend with the requested data and, finally, the frontend displays the data in a series of graphs and charts to the user.

In other words, the structure of the CITYSENT system is highly modular. While designing the system, we focused on modularity because it allows components to easily be extended with new features without breaking any of the other components, which was an important wish of the clients.

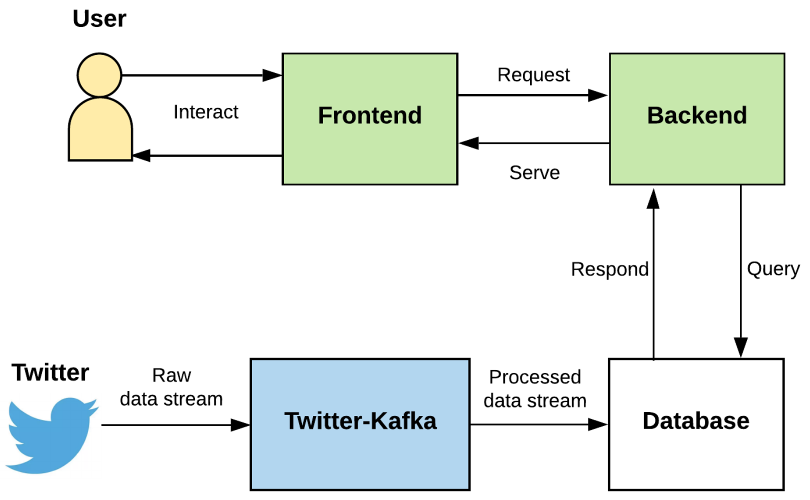


Figure 2.1: The structure of CITYSENT. The green components make up the web application and the blue represents the stream processor. The arrows show in general terms how the various components interact with each other and how the data is collected from Twitter, analysed and finally presented to the end-user.

2.2. Stream processing

At the very beginning of the project, we were faced with a difficult design choice that would shape the rest of the project: how exactly do we obtain data from social media?

When gathering data from social media there are two options: using the platform's own API or scraping. Each comes with its own set of problems. Using the API is a lot more convenient than scraping since the retrieved data is already parsed into a (semi-)structured format. However, access to APIs has become increasingly more restricted in light of privacy concerns and the Cambridge Analytica scandal [1]. On the other hand, scraping is both difficult to maintain and a topic of active legal and ethical debate [2]–[4]. The problematic and time-consuming nature of implementing and/or integrating a scraper makes scraping infeasible for the scope of this project. We therefore chose to use Twitter's API, since it is the least restrictive out of the big social media platforms (e.g. Facebook, Instagram). It also provides us with valuable geographical information related to the tweets, which is important when analysing sentiment related to geographical locations.

Twitter has a wide range of APIs available for different use cases, including free and paid options [5]. We first considered the Twitter Standard Search API, but with this API we could only fetch tweets from the past seven days. This caused a problem since we wanted a sizable collection of historic tweet data. A possible solution to this problem was to let the API run every once in a while so that we could collect a useful database over time. This led to even more questions: when would we run the data collection? Once a week? At night? Setting a timer or fixed time for the system to collect tweets would be possible, albeit awkward and outdated. Instead, we came up with a better solution: Twitter's Stream API. For continuous collection of data, stream processing is a natural solution. The free version of the Stream API allows us to access a stream of a random sample of 1% of all tweets matching our search criteria–in our case tweets originating from Belgium and The Netherlands. This allows us to keep our database constantly up-to-date, distribute the workload needed to process tweets throughout the day and allows the dashboard to give an accurate depiction of both historic and real-time data.

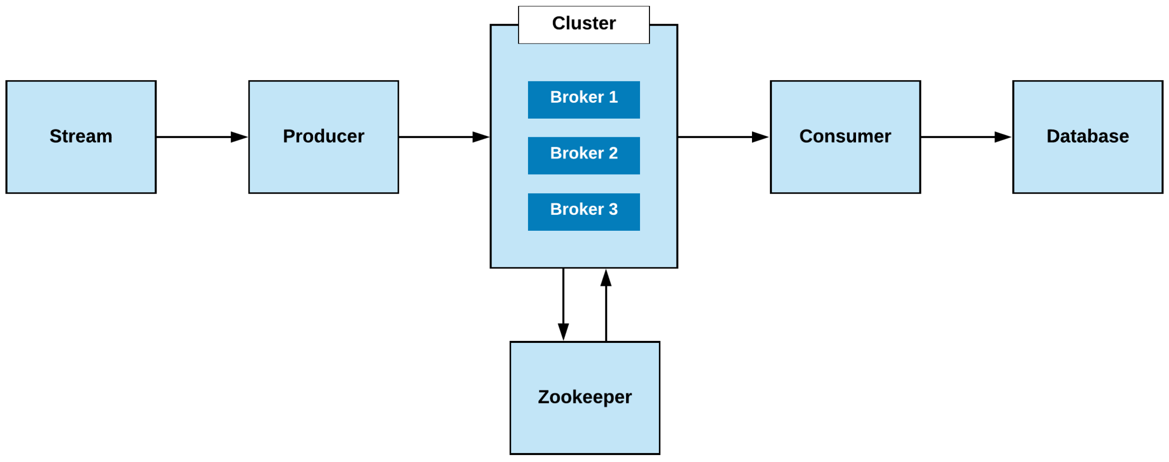


Figure 2.2: Example Kafka structure

Apache Kafka is an open-source distributed streaming platform [6] and a durable publish-subscribe messaging system developed by LinkedIn. There is a lot to Apache Kafka, but a simplified example of the Apache Kafka system structure can be seen in figure 2.2. The *producer* connects to a data stream and publishes the data it receives to a *topic* within a *cluster*. A topic is an append-only log of data records. They reside in *brokers* within clusters, see figure 2.3. The *consumer* subscribes to a topic, and reads the log of data in sequential order, see figure 2.4. *Zookeeper* ensures that when a consumer dies, its progress is saved such that another (or the same) consumer can pick up where the other left off. This architecture is highly scalable, as additional producers and consumers can be added without having to change the system. The cluster is what makes Apache Kafka a distributed system; the brokers can be multiple computers belonging to the same network of computers. This means that your data is spread over several nodes in the network so that even if one node goes down, your data is still available.

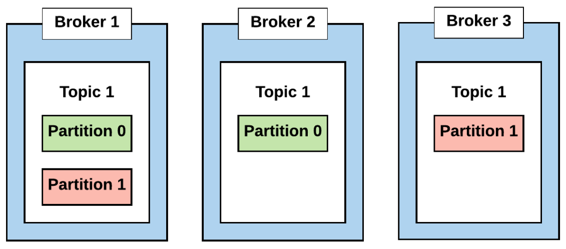


Figure 2.3: A topic is partitioned and copied across brokers to provide fault-tolerance.

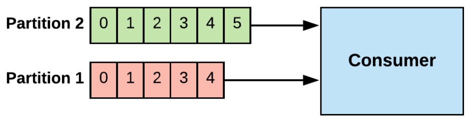


Figure 2.4: The consumer subscribes to a topic of a broker and reads it sequentially.

It is designed to be fast, fault-tolerant and scalable—which are all attributes you want in a stream processing system [7]. We chose Kafka because it is well-documented and a popular streaming solution used by several well-known companies handling big data such as Twitter [8], Netflix [9] and Zalando [10]. It does everything we need it to—collect and process streaming data—while also providing guarantees like high-throughput and horizontal scaling. High-throughput is important so that the data collection pipeline does not get clogged. Given enough funds, the system could also be extended to use a more powerful API that delivers more data. Horizontal scaling is important because of the modular and expandable nature of CITYSENT. Using Apache Kafka, CITYSENT can easily be extended to collect data from multiple social media sources, or expanded to a network to introduce fault-tolerance and the ability to handle heavier data loads.

For the scope of this project, the Apache Kafka structure is rather simple, using only one broker since we only have access to one server. However, as shown in 3.1.1, Apache Kafka works well at small scales as well and CITYSENT has the potential to scale considerably given more funds and access to hardware.

2.3. Web application

CITYSENT is accessible through a web application available at https://citysent.ewi.tudelft.nl. This is done so that it is easy to use the tool, without the need to download anything. However, web applications are not without security risk. Therefore, in addition to making the tool easily accessible, we have also focused on making it secure to use by putting a couple of security measures in place.

Firstly, all requests outside of the TU Delft will be ignored. This is done to ensure that no malicious users from outside the TU Delft have access to the tool. If a user has a VPN and can connect to the VPN of the TU Delft, access from other places is also possible. Secondly, we have made sure that users always have a secure connection to the application using HTTPS and TLS 1.3 encryption. Although the data we send and receive from the users is public information and therefore not particularly exploitable, our site could still be an easy target for man-in-the-middle (MITM) attacks. Without HTTPS and a valid TLS certificate, the user cannot be sure whether our site is authentic or a malicious front. Implementing HTTPS lowers the user risk as it becomes harder compared to HTTP to successfully perform MITM attacks [11].

3 Implementation

3.1. Stream processing

3.1.1. Twitter-Kafka

An overview of the structure and the different components within Twitter-Kafka can be seen in figure 3.1. The backbone of the Twitter-Kafka system is the Twitterproducer and Twitterconsumer. These were implemented as separate modules and act independently of each other to take advantage of the horizontal scaling capabilities of Kafka. They can be run as separate processes on a machine, or even on different machines entirely. The Kafka environment used by Twitter-Kafka is simple: it consists of one cluster and one broker containing a single topic where intermediary tweet data is temporarily stored before analysis. The reason for this simple structure is that we only had one server at our disposal, meaning that the system is less fault-tolerant than it could be if we had access to a distributed network of computers. Nevertheless, the amount of data that comes in from the free version of the Twitter Stream API is easily handled by one machine alone. The simple structure fits the scope of this project while also allowing for considerable expansion in the future.

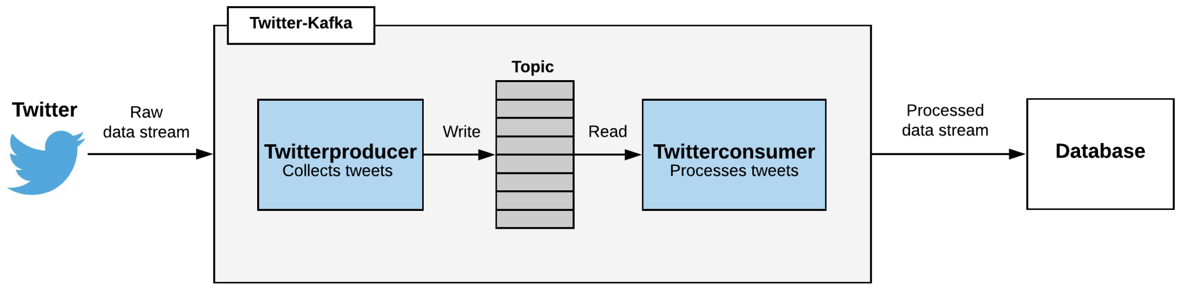


Figure 3.1: Twitter-Kafka overview

The Twitterproducer connects to the raw data stream of tweets that is provided by the Stream API. The approximate bounding boxes of Belgium and The Netherlands are used as filter criteria. However, because countries are not perfectly square, the Twitterproducer will also receive some tweets from the nearby countries Germany and France that are close to the Dutch and Belgian borders. The raw tweet data also contains some irrelevant and potentially sensitive information (like usernames) that we are not interested in for sentiment analysis. The job of the Twitterproducer is therefore more than just collecting tweets; it also cleans the data by discarding irrelevant tweets and information not used in the data analysis. The cleaned (or intermediary) data is then temporarily stored in a topic.

Storing intermediary data in the topic is useful because it allows the Twitterproducer and Twitterconsumer to act completely independently of each other. The topic acts as a queue of messages that the Twitterconsumer reads from. The queue might fill up when tweets arrive faster than they can be processed at peak Twitter activity, and empty when there is less activity. Within this topic, profile picture links and tweet text is stored (among other data). Both of these pieces of information are sensitive, and Twitter's Terms of Service does not allow for long-term storage of tweet content in case the author modifies or deletes the original content. The temporary nature of the topic is therefore perfect, as this sensitive information is deleted after 48 hours.

The Twitterconsumer is the last step of the data collection pipeline. It reads the cleaned tweet data from the topic and performs a series of analyses: textual analysis in the form of sentiment and keyword analysis, as well as image analysis to estimate the age and gender of the author of a tweet. Once all analysis is done, it anonymizes the data before storing it in the database. The data is anonymized by discarding sensitive information like profile picture, tweet text and generalizing the timestamp to only be hourly.

3.1.2. Textual analysis

As part of the Twitterconsumer, tweets that stream into the system need to be processed. This is done in several stages. First of all, tweets are preprocessed to clean them and prepare them to be analysed at a later stage. Next, the preprocessed tweets are used as input for the sentiment analyser, which determines the sentiment class of tweets and the set of keywords that describe these tweets.

Preprocessing

Preprocessing of tweets is done in several stages. First of all, the analysis of tweets requires them to be in English. Therefore, Dutch tweets are translated into English if necessary. This is done using the Googletrans API [12], which allows for the automatic translation of text between a large set of languages. Googletrans itself makes use of the Google Translate Ajax API [13] to achieve this.

Then, tweets are cleaned of smileys, user-mentions and some Twitter-specific keywords, such as "RT". User-mentions and Twitter keywords namely do not contain any sentimental value, making them negligible to sentiment analysis. These words can even have a negative influence on the accuracy of sentiment analysis because they add noise to tweets. Although they can contain sentimental data, smileys and emojis are also removed from tweets in this stage. The reason for this is explained in the section about sentiment analysis.

Next, all letters in tweets are converted to lowercase to standardise the words in these tweets. Finally, punctuation is removed and tweets are stemmed, which is a technique that replaces different variations of words by their basic variant (stem). For example, 'greatly' and 'greatest' are replaced by 'great'. This technique is another method to standardize words in tweets.

These techniques were selected from a larger set of techniques. The selection is based on the extensive literature study described in the research report (Appendix E). Techniques that were not selected include spelling correction, slang and abbreviation normalization, because these did not show signs of significant improvements in the accuracy of sentiment analysis in the existing literature. A technique that was selected during research, but was not used in the final product is negation handling, because this technique did not show a significant accuracy improvement of sentiment analysis in practice.

Sentiment analysis

After being preprocessed, tweets need to be analysed to determine the sentiments they express. Sentiments are represented by three classes: positive, neutral and negative. The analysis is done in two stages.

The first stage is lexicon-based analysis. In lexicon-based sentiment analysis, lists of words annotated by sentiment scores, also called polarity scores, are used to determine the sentiment of a piece of text. This can be done by summing the sentiment scores of the words present in the text such that the result indicates the overall sentiment score of the text [14]. The annotated words can be extended by the use of other elements such as annotated emoticons and several linguistic rules [15].

The main benefit of lexicon-based analysis is that it is unsupervised, meaning that once a lexicon is obtained, no training data is required to train the system [15]. A downside, however, is that the language used in tweets is dynamic, as new expressions are constantly emerging [15], which makes it hard to have an up-to-date and accurate lexicon. Furthermore, the fact that polarity scores of text are calculated using the polarity scores of single words causes another problem, namely the impossibility of taking the context of words into account. This makes it hard to obtain a valid sentiment score from ironic statements or other statements that heavily rely on context.

Lexicon-based analysis is used to determine whether the sentiment score of a tweet points to neutrality. If this is the case, the tweet is classified as neutral. If this is not the case, the tweet is further analysed in the second analysis stage.

The second stage is the usage of a machine learning model. For this stage, a supervised learning algorithm is trained to classify pieces of text into several discrete sentiment categories. After training, the model can be used to predict whether the sentiment of an input tweet is positive or negative.

For the classification, several models were taken into account: Decision Trees, Naive Bayes and Support Vector Machines.

Decision trees are classifiers constructed as a tree: in these trees, all non-leaf nodes represent decision points [16]. The branches that originate from these decision points represent the outcomes belonging to these points and the leaf nodes represent the class labels of the classifier.

The benefits of decision trees are that they are often relatively simple to interpret and that they can be used very well to perform classification operations on text. However, decision trees can also easily be overfitted on data [14].

Next, Naive Bayes is a probabilistic classification model based on the Bayes theorem [17]. Using this theorem, several class labels, or hypotheses, can be defined. For every problem instance that needs to be classified, the probability of the instance belonging to a certain class label can be determined. The instance is then predicted to belong to the class with the highest probability.

The Naive Bayes approach is often used in sentiment analysis for its simplicity and effectiveness [14].

Finally, Support Vector Machines (SVMs) are used to find hyperplanes in N-dimensional spaces, where N is the number of features, such that the data points belonging to separate classes are separated by the hyperplane [18]. The goal of SVMs is to find the hyperplane, such that the margin (i.e. the distance between the two closest data points of the separate classes) is maximized to improve the confidence of predictions.

This approach has often been used in text classification scenarios where it was very successful, as SVMs can operate on large feature spaces. The downside of SVMs, however, is that they are not simple to understand and computationally expensive [14].

Existing literature indicated that Support Vector Machines are generally more accurate for text analysis tasks than the other considered models. Their computational expensiveness is not seen as a major downside, as the model will be deployed as part of a streaming system, meaning that only a small amount of subsequent data need to be processed.

A more detailed explanation of the characteristics, downsides and benefits of the different models can be found in Appendix E.

Besides the models already covered in E, an SVM-based model was also compared to a linear regression-based (LR) model. Both models were implemented using Scikit-learn [19] and trained on twitter data. Their accuracies for different regularization parameters (C) were then determined. The results of this can be seen in 3.1

Table 3.1: Accuracies for an SVM and an LR-based model for a variety of regularization parameters.

C/model	SVM	LR
0.001	0.8080	0.7794
0.005	0.8199	0.7998
0.01	0.8233	0.8083
0.05	0.8241	0.8200

This table shows that the SVM-based model performs slightly better than LR-based model. Therefore, an SVM is chosen as the used classification model in CITYSENT.

To train the SVM, a data set is needed in which tweets are annotated by the sentiments they carry. The data set that is chosen for this, is the sentiment140 data set [20], because this data set, created by researchers of Stanford University, contains 1.6 million tweets, annotated by their sentiment class.

It would also have been possible to use the machine learning model for all three sentiment classes, instead of using it for only positive and negative. This would have removed the requirement of lexicon-based analysis. However, the data set that was used to train the model, sentiment140, only contains positive and negative tweets. As not many other open access data sets are available that have a similar quality and size as sentiment140, this would have required the manual annotation of a large number of neutral tweets. This was seen as a wrong investment of time for the scale of this project. Therefore, it was chosen to use lexicon-based analysis for determining tweet neutrality.

Similarly, the sentiment140 data set contained no emojis and emoticons. Therefore, they are not taken into account during sentiment analysis.

Keyword generation

During processing, keywords of tweets are also determined. In order to do this, the importance of individual words within a tweet needs to be determined. In CITYSENT, such importance is measured according to the TF-IDF method. A score is assigned to every word in a tweet according to the following formula:

$$tfidf_{i,j} = tf_{i,j} \cdot \log(N/df_i)$$

Where, $tf_{i,j}$ is the number of occurrences of a word i in a tweet j, df_i the number of tweets in the sentiment140 data set containing word i and N the total amount of tweets in the sentiment140 data set.

The words in a tweet are ranked according to this measure. And a maximum of three words with the largest scores is selected as keywords.

Stopwords, commonly used words like ‘a’ and ‘the’, are not taken into account during the TF-IDF calculation.

3.1.3. Face recognition

When we first deployed the sentiment analysis, we found that some small places were extremely negative and had a lot of tweets. We found out that this was caused by weather stations tweeting the weather every 30 minutes. If the weather was bad, our system would interpret the tweet as negative. This is of course not data we want to include and thus we decided to discard every tweet that has no face or more than 5 faces in the profile picture. This way, most companies and political parties get discarded, since they often don’t have a face as profile picture. Some people might have a group photo as profile picture, this is why we allow up to 4 faces in a picture.

Face recognition is done with a pre-trained face recognition model called face_recognition. According to its Github page, the accuracy is 99.38%. Local tests indicated that this is correct.

3.1.4. Age & gender

Since Twitter does not share age nor gender, CITYSENT has to infer it from profile pictures. This inference is done with the following pipeline:

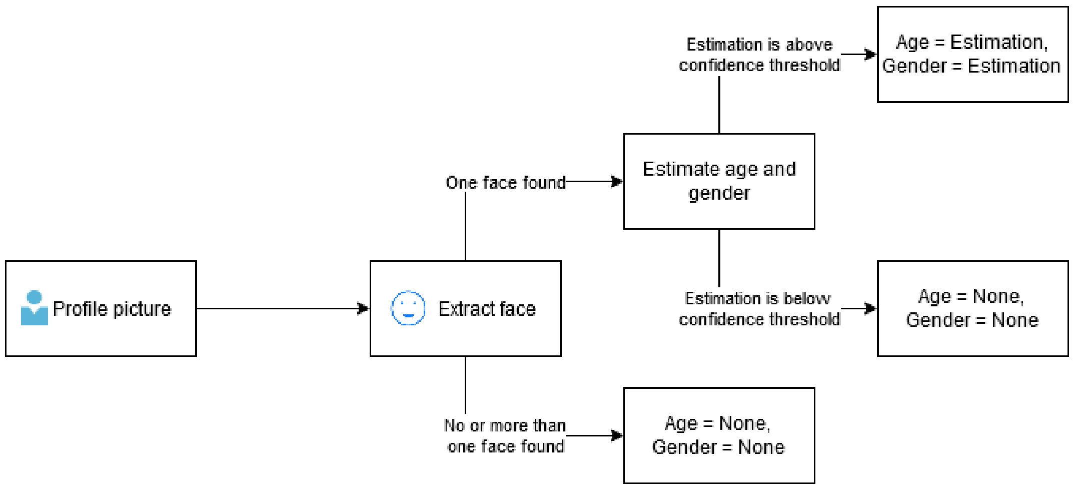

Profile picture
Extract face
One face found
Estimate age and gender
Estimation is above confidence threshold
Age = Estimation, Gender = Estimation
Estimation is below confidence threshold
Age = None, Gender = None
No or more than one face found
Age = None, Gender = None

Figure 3.2: Age and gender estimation pipeline

Age and gender estimation is done with a Convolutional Neural Network, which is ”a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other" [21]. Because of the time constraints and scope of the project, the decision was made to not implement such a neural network from scratch, but to look for already existing models. It was decided to use a CNN model created by Gil Levi and Tall Hassner [22]. This model was chosen, because it has high accuracy, but is also straightforward to implement. How the neural network should be initialized, was based on a tutorial from LearnOpenCV [23]. While this tutorial was a very good starting point for age and gender estimations, the following three processes have been added to increase the accuracy:

Face cropping
Face cropping removes the noise of an image and allows the age and gender estimator to only process the face, not the surroundings. The OpenCV tutorial already implemented face cropping, but the accuracy of that method proved to be insufficient during validation tests. Sometimes, non-existing faces would be recognized, which would then lead to estimations being done on invalid cropped images. This is why we decided to do the extraction of a face with the same face recognition tool as mentioned in 3.1.3.

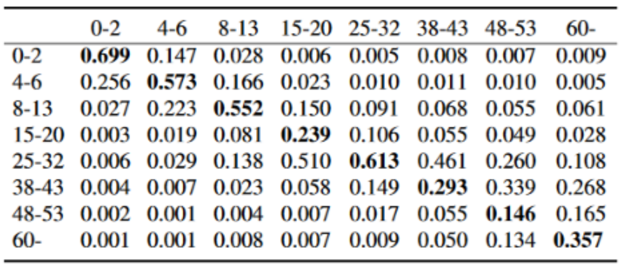
According to Levi and Hassner, face cropping improves gender accuracy with 8% and age accuracy with 5%, which is in line with our own findings.

Confidence threshold
Every estimation is based on an array of confidences (from 0 to 1). For each age, a certain confidence is given and the highest confidence will be the estimated age. This does mean that if every confidence is very low, the estimation is probably not very accurate. Because of this, CITYSENT will compare the biggest confidence to a certain threshold value. If the confidence is lower than the threshold, the estimation is discarded. This will lower the number of estimations, but will increase accuracy. The threshold has been determined through manual testing and tweaking. For age estimations, the category confidence threshold is 0.95 and the bracket confidence threshold is 0.98. For gender estimations, the threshold is 0.9. These thresholds can always be changed, according to the needs of the clients, but we feel these thresholds give a good balance between accuracy and quantity. Because we have 1500 lines of data logging for different thresholds, appendix C contains only relevant thresholds and their accuracies.

Age brackets and categories
The model made by Hassner and Levi return the following age brackets: {0-2}, {4-6}, {8-13}, {15-20}, {25-32}, {38-42}, {48-53} and {60+}. There are gaps between these brackets to make the estimation easier. Take for example (8-12) and (15-20). It is very easy to see that someone who looks 14-15 belongs to (15-20), because they are clearly not (8-12). Now, if you have (8-14) and (15-20), confusion will be very high for this person.

In their report, Hassner and Levi put a confusion matrix, showing the accuracies of their model [22]:

Table 3.2: Confusion matrix made by Levi and Hassner (top is actual age, left is predicted age)



	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60-
0-2	**0.699**	0.147	0.028	0.006	0.005	0.008	0.007	0.009
4-6	0.256	**0.573**	0.166	0.023	0.010	0.011	0.010	0.005
8-13	0.027	0.223	**0.552**	0.150	0.091	0.068	0.055	0.061
15-20	0.003	0.019	0.081	**0.239**	0.106	0.055	0.049	0.028
25-32	0.006	0.029	0.138	0.510	**0.613**	0.461	0.260	0.108
38-43	0.004	0.007	0.023	0.058	0.149	**0.293**	0.339	0.268
48-53	0.002	0.001	0.004	0.007	0.017	0.055	**0.146**	0.165
60-	0.001	0.001	0.008	0.007	0.009	0.050	0.134	**0.357**

According to this confusion matrix, the hardest ages to estimate are {15-20}, {38-42} and {48-53}. We tried to fix this by combining certain brackets and creating categories. As can be seen in the following confusion matrix:

Table 3.3: Confusion matrix made with 3 categories: child {0-13}, adult {15-53} and senior {60+}

	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60+
Child	**0,982**	**0,943**	**0,746**	0,179	0,106	0,087	0,072	0,075
Adult	0,015	0,056	0,246	**0,814**	**0,885**	**0,864**	**0,794**	0,569
Senior	0,001	0,001	0,008	0,007	0,009	0,050	0,134	**0,357**

Using three categories has its trade-offs; correctness increases at the expense of accuracy. For example, a person of age 15 will probably behave a lot differently than an adult of 32, but they both get classified as 'adult'. Because of this, CITYSENT gives the user the ability to toggle between age brackets or categories.

The categories are calculated by combining all the brackets of a certain category and then finding the max confidence. This could, in rare cases, cause the brackets to not be in line with the category estimation. As an example are these confidences:

Table 3.4: Confidence array for brackets

Bracket	(0-2)	(4-6)	(8-13)	(15-20)	(25-32)	(38-43)	(48-53)	(60+)
Confidence	0.0025	0.01	0.96	0.01	0.005	0.005	0.005	0.0025

Here, {8-13} has the highest confidence, but will not be chosen, since the confidence is below the bracket confidence threshold of 0.98.

Table 3.5: Confidence array for categories

Category	Child	Adult	Senior
Confidence	0.9725	0.025	0.0025

If the brackets are combined into categories, 'Child' will be chosen, since that category has the highest confidence and the confidence is above the 0.95 threshold for categories.

The database will thus in this instance contain a tweet that has as age bracket 'None' but as age category 'Child'. This issue could be fixed by having an age bracket confidence lower than category confidence, but after a lot of tests, we believe having these thresholds give the most optimal balance between bracket discarding and estimation accuracy.

3.2. REST API

To connect the front-end to the back-end, we implemented a REST API to serve all the data. The API provides easy access to all the data that is displayed in the user interface. To actually retrieve the data, the back-end queries the database. Most of the API endpoints each have their own query to retrieve the corresponding data and perform some simple processing of the data to create a JSON response in a convenient format. Each endpoint also accepts certain filters to apply to the data. Because we wanted to visualise various aspects of the data, it was important that the queries could be generated dynamically. This allows us to easily query different dimensions of the data set using the same endpoint.

Applying the filters dynamically was still a challenge, because we wanted to support filtering on different variables of different types, using different logical structures. At the same time security needed to be kept in mind to ensure that SQL injection was not possible, as any user could supply these filters in a request to the API. To make these dynamic filters possible we created a query builder that has a template for each query and generates the where clause based on the supplied filters. To deal with the different types of filters, all possible filter options are specified in this query builder with their corresponding SQL, making it possible to for example compare dates and locations in different manners. For dates, you could supply a filter saying "greater than this start date", while for locations you could supply a filter saying "inside of this area". We avoided the vulnerability of SQL injection by generating parameterised SQL rather than inserting the values of the filters directly.

3.3. User Interface

In the first week of the Bachelor End Project, the scope of our project was creating a valuable data set containing the sentiment of people towards certain places and cities. To deliver a complete product we decided to create a research tool for the clients, in the form of an interactive dashboard of the collected data. The dashboard will contribute to better analysis of the data and sentiment. Nevertheless, we have still respected the clients' initial wish of a data set by adding a download feature, allowing the data set used by the dashboard to be downloaded as a CSV file. Our aim was to create a tool accessible from every computer, so research could be done from anywhere, anytime, without downloading a program on your computer. For this reason, we decided to create a website. For the creation of the user interface, we decided to use React since it enables you to easily create reusable components. In our dashboard design, we reuse a lot of components: the 'cards' the charts are on, the left and right half of the analysis screen, the doughnut charts, and more.

3.3.1. Design choices

In the first sprint, we had a meeting with the clients to discuss what should be included in the tool. As preparation for this meeting, we drew a mock-up of what we thought should be included in the dashboard. In figure 3.3 you can see the mock-up we created for this first meeting. After discussing it with the clients we concluded the following things were missing:

- Filter the sentiment by gender
- Filter the sentiment by age group
- Filter on positive or negative sentiment
- Filter on certain sentiment words
- Filter on different dates at the same location
- Filter on different locations on the same date
- Download/export button
- Accuracy of age and gender prediction
- Display male, female AND unknown genders
- Display child, adult, senior AND unknown age

Considering these new insights into the clients' desires we drew new mock-ups. We decided to split the tool into four different pages:

1. Overview (Home page): an overview dashboard where we add components displaying information about the full data set. (figure 3.5)
2. Analysis: a split screen where the end-user can apply filters to the data set and the dashboard components change accordingly. Due to the split screen, the end-user can compare different filters, such as a different date at the same location or a different location on the same date. (figure 3.6)
3. Map: in the map, the end-user can search for a specific location and the sentiment and amount of tweets it is based on is shown. At first, we wanted to implement a feature in the map so the sentiment is displayed over the full map as an overview. This turned out to be too much work for the time we had available. (figure 3.7)
4. Information: in the information page we added information about how we derived the data we are displaying and their accuracy's. (figure 3.7)

For the creation of the dashboard we used the 'Devias Kit' Dashboard.

The colors we used in our dashboard were based on ColorBrewer. This is a website that helps you select colors for your application and thanks to ColorBrewer we have created a colorblind-friendly tool.

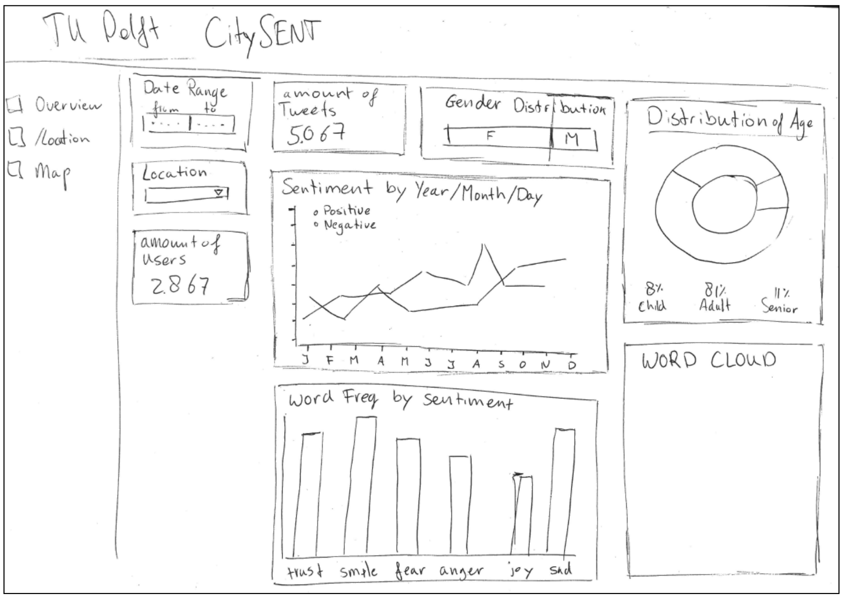


Figure 3.3: First dashboard mock-up

3.3.2. Changes in design during implementation

For the overview/home page, we did not change much compared to the mock-up. As shown in figure 3.5, the main changes are the elimination of the ’amount of users card’ and the ’word frequency by sentiment card’. The ’amount of users card’ was left out because including the number of individual users in our database meant implementing a lot of security measures. This was not feasible within the project’s scope. In section 9.2.7 we elaborate more on this. The ’word frequency by sentiment card’ was left out because we integrated this functionality directly into the word cloud. The word cloud uses font size and color to display the most frequently used words and their sentiment. When you hover over the words, the exact amount of usages in tweets will show.

For the Analysis page, as shown in figure 3.6, the main changes are the elimination of filtering on sentiment words, the addition of the ’sentiment score over time graph’, the addition of the ’number of tweets per sentiment over time graph’ and as explained above also the deletion of the ’amount of users card’ and the ’word frequency by sentiment card’. We left out filtering on keywords because it was not feasible to do within the scope of this project. We do think this would be a nice improvement in the future, however. Furthermore, we added two graphs to the Analysis screen, namely the ’sentiment score over time graph’ and the ’number of tweets per sentiment over time graph’. These graphs were included to give insight into the sentiment over time, rather than just two specific data sets.

3.3.3. Problems during implementations

During the implementation of the interface we did not face major issues, but mostly small problems; problems that sound like a very small fix, but in reality could take hours. For example, we had trouble with uploading the TU Delft logo. It turned out that images can only be locally imported if they are in the static folder. Tiny problems like this occurred, because none of us had any real experience with web development.

Another problem we faced was how long it took for the interface to run. In web development, it is possible that your code still compiles even though there is a small mistake that causes the page not to render. Often we had to wait a minute or longer for the web page to load, only to find out it was broken. Even if the code did not break, the waiting time only became longer as our project got bigger. At this moment, the efficiency of our user interface implementation was low. To solve this, we started using the ”watch” functionality of Webpack, which checks your files for changes and updates accordingly, rather than having to rerun the full project. This saved us a lot of valuable time.

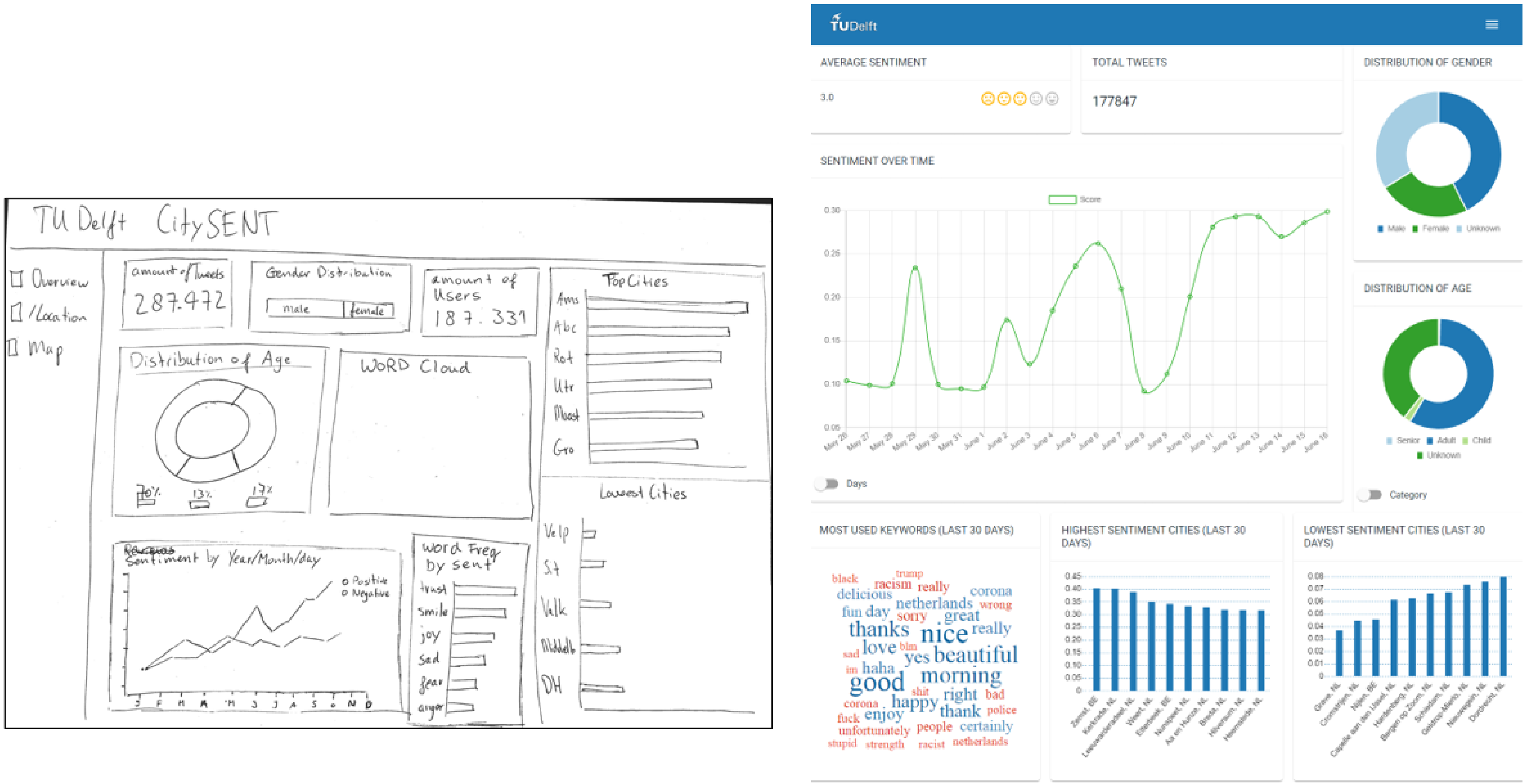


Figure 3.4: Left: Second dashboard mock-up | Right: Final Dashboard page

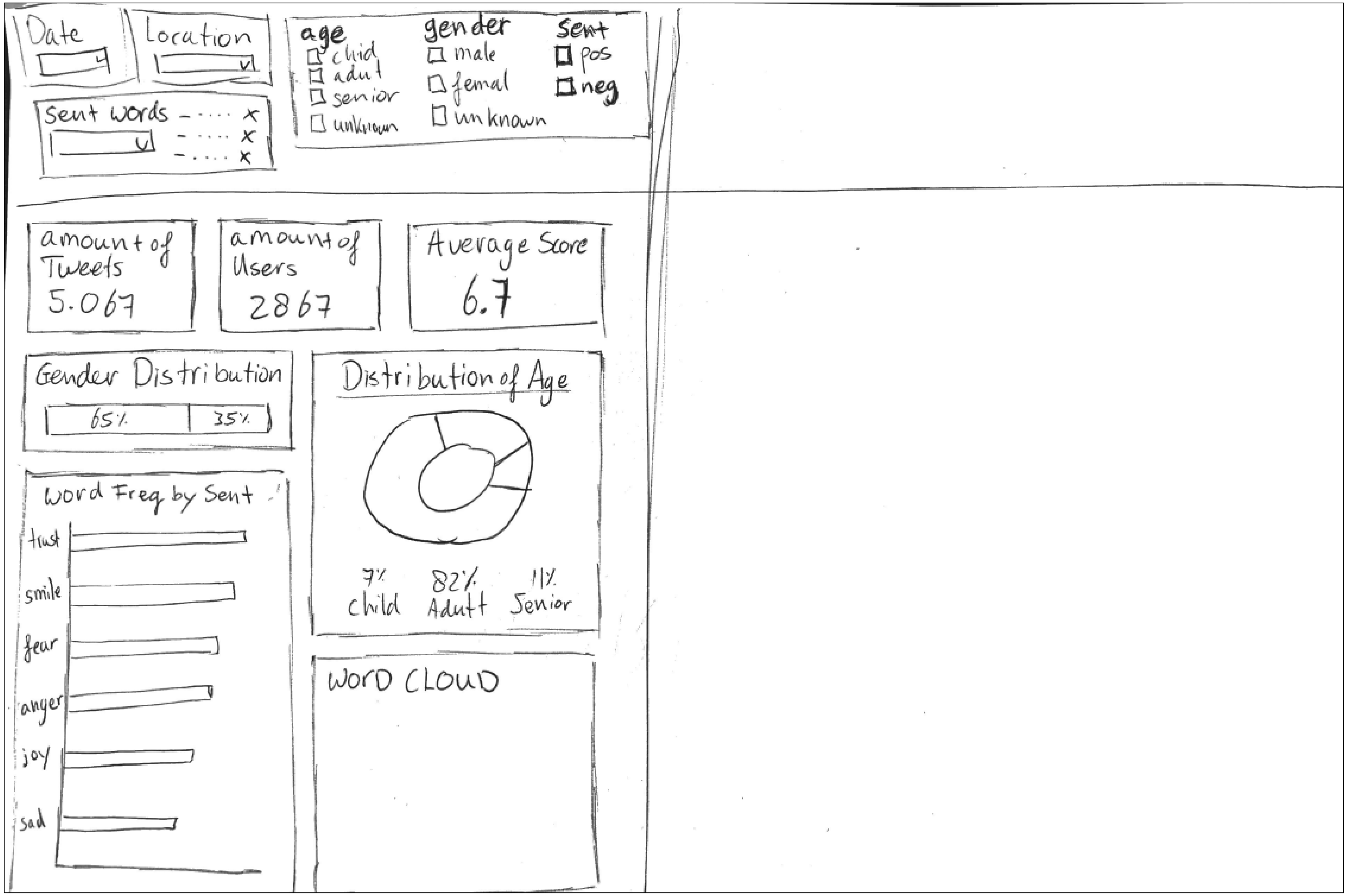


Figure 3.5: Analysis page mock-up

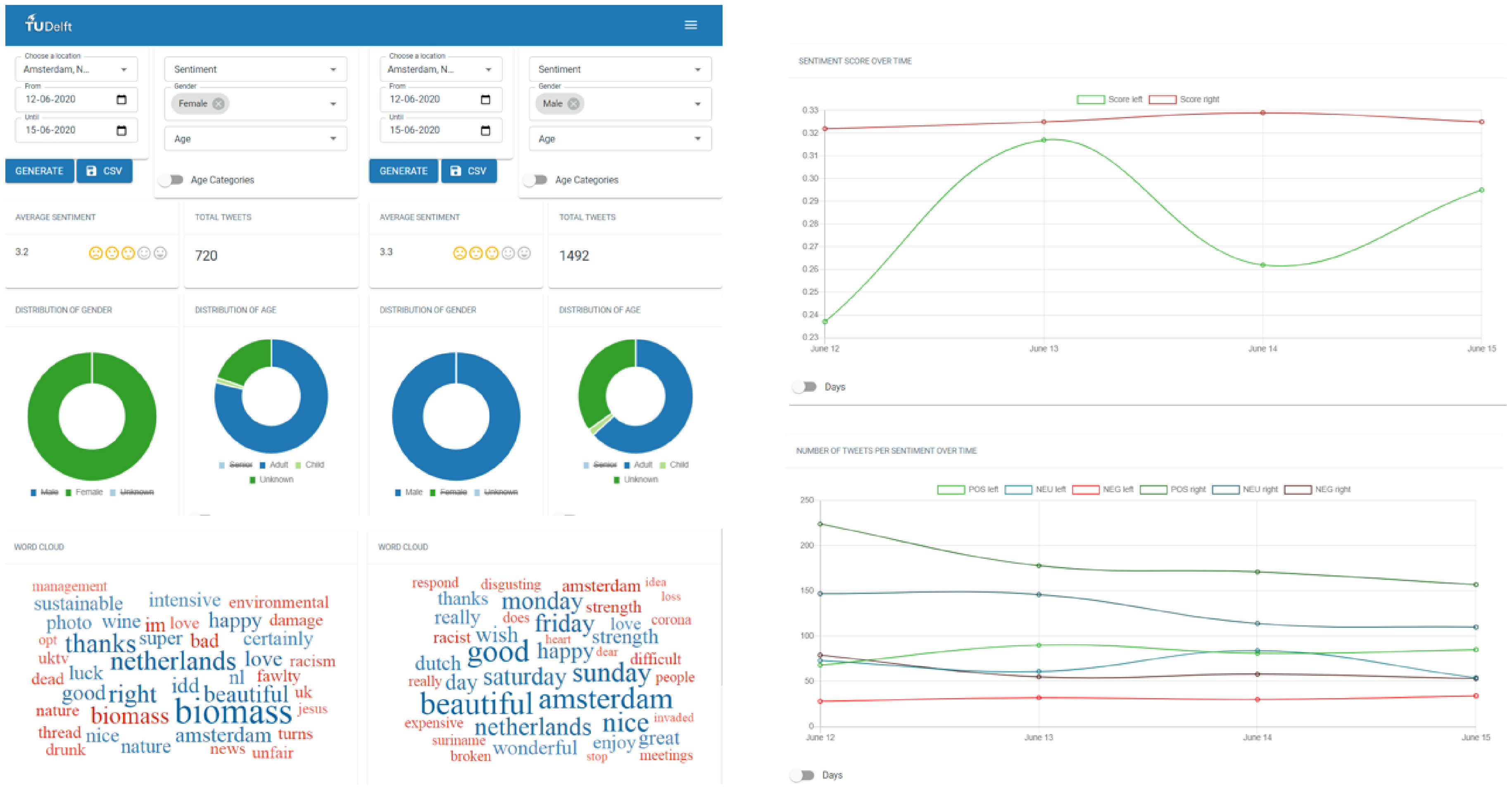
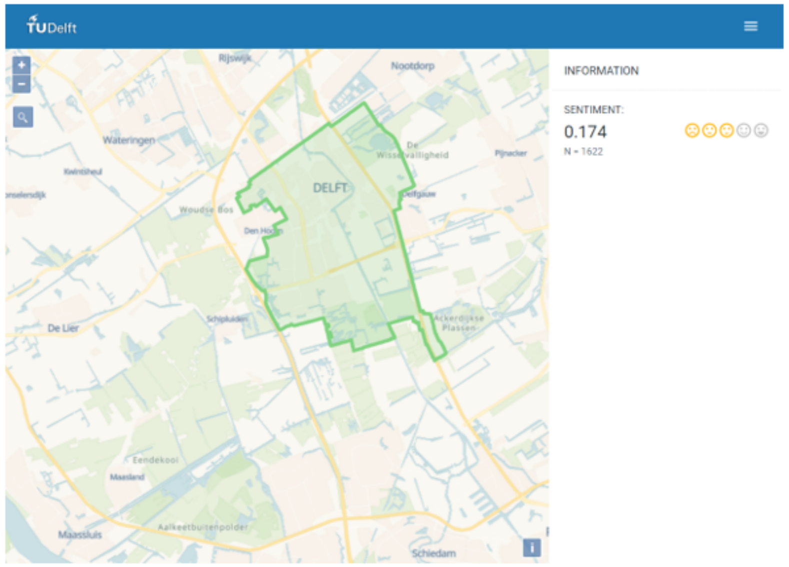


Figure 3.6: Left: Top half of the final Analysis page | Right: Bottom half of the final Analysis page

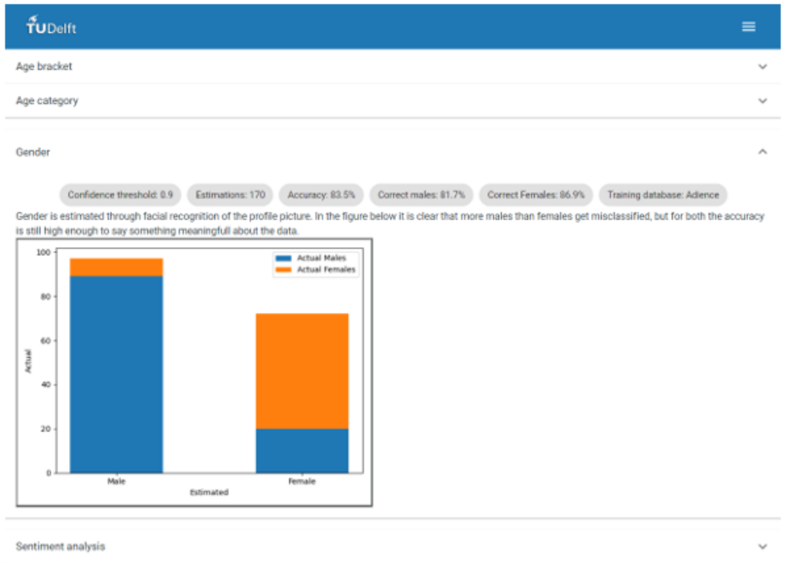


INFORMATION

SENTIMENT:

0.174

N = 1622



Age bracket

Age category

Gender

Confidence threshold: 0.9 | Estimations: 170 | Accuracy: 83.5% | Correct males: 81.7% | Correct Females: 86.9% | Training database: Adience

Gender is estimated through facial recognition of the profile picture. In the figure below it is clear that more males than females get misclassified, but for both the accuracy is still high enough to say something meaningfull about the data.

Sentiment analysis

Figure 3.7: Left: Final Map page | Right: Final Information page

4 Accuracy

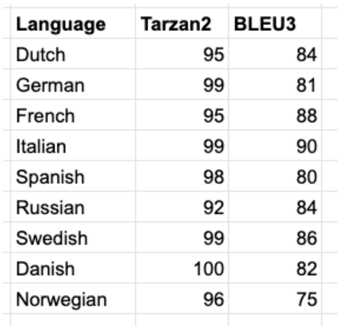
A big part of CITYSENT is based upon machine learning. While machine learning has been proven to be very useful to CITYSENT, it does bring some uncertainties to the data. Machine learning is fundamentally an algorithm that makes predictions based on previous data. The problem is that predictions are always uncertain and can thus never be regarded as absolute truths. This poses a problem for CITYSENT, because this tool's goal is to provide sentiment data to researchers. But how can researchers be sure that the data they are provided with is correct? They can't. This is why transparency is of the utmost importance in their research. This chapter aims to provide the information needed for researchers using CITYSENT, to determine the accuracy of their research. The following modules are discussed in order of when they are used in the system: tweet translation, sentiment analysis, age prediction and gender prediction. Another aspect that is discussed in this chapter is location filtering. While this aspect does not make use of machine learning, it does contain slight inaccuracies that are important to be aware of.

4.1. Tweet translation

As stated in 3.1.2, Dutch tweets are translated into English before the sentiment analysis. This translation can negatively influence accuracy if wrong translations are made. Luckily, we are using Google Translate, which is capable of doing translations with high accuracy.

In 2019, a study was conducted about the accuracy of Google Translate [24]. In table 4.1 a table is presented of multiple languages and their accuracy score on Google Translate. BLEU (Bilingual Evaluation Understudy) is an accuracy evaluation method for translations. "BLEU looks at the presence or absence of particular words, as well as the ordering and the degree of distortion—how much they are separated in the output" [25]. BLEU3 looks at the accuracy of translating foreign languages to English with Google Translate in 2019. The Tarzan score is the percentage of text that can be understood by humans. Tarzan2 also looks at the accuracy of translating foreign languages to English with Google Translate in 2019.

Table 4.1: Google Translate accuracy score [24]



Language	Tarzan2	BLEU3
Dutch	95	84
German	99	81
French	95	88
Italian	99	90
Spanish	98	80
Russian	92	84
Swedish	99	86
Danish	100	82
Norwegian	96	75

Looking at the table, it is clear that the translation from Dutch to English is very good. Also, as of the writing this report, no translation has been found that changed the sentiment of a sentence. So it is safe to say the impact of translating on sentiment is very minimal.

4.2. Sentiment analysis

The accuracy of the SVM used in CITYSENT is determined by splitting the sentiment140 data set into three parts: a training set, a testing set and a verification set. The training set is used to train the SVM, the validation set is used after each training epoch to determine whether the model is converging. Finally, the testing set is used once to determine the accuracy of the SVM when it is provided with unseen data. This approach shows an accuracy of 82,4%.

In figure 4.1, a confusion matrix is shown that shows the rate of the tweets that are correctly classified, compared to those that are not. It should be noted that this confusion matrix was created using only a subset of the total training data, such that the overall accuracy shown in this figure is smaller than the 82,4% accuracy of CITYSENT. In the figure, negative class labels are represented by 0 and positive class labels by 4. This shows that negative tweets are wrongly classified slightly more often than positive tweets.

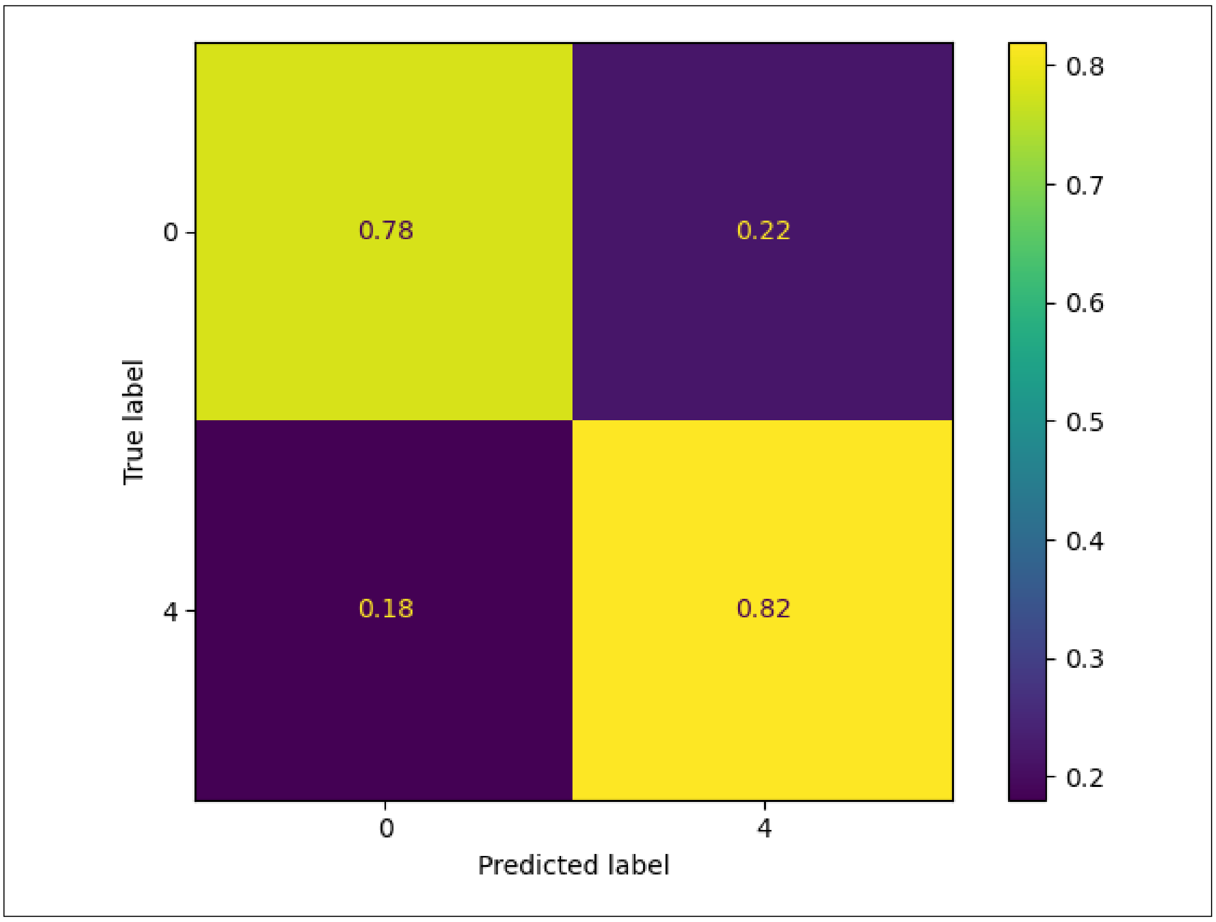


Figure 4.1: Relative amount of tweets that are correctly and incorrectly classified.

4.3. Age and gender prediction

As stated in 3.1.4, the inferrence of age and gender is done with a Convolutional Neural Network, based on the work of Gil Levi and Tall Hassner [22]. Although this model is celebrated among researchers, it is not 100% accurate. The following accuracies have been reported on 477 tweets[1]:

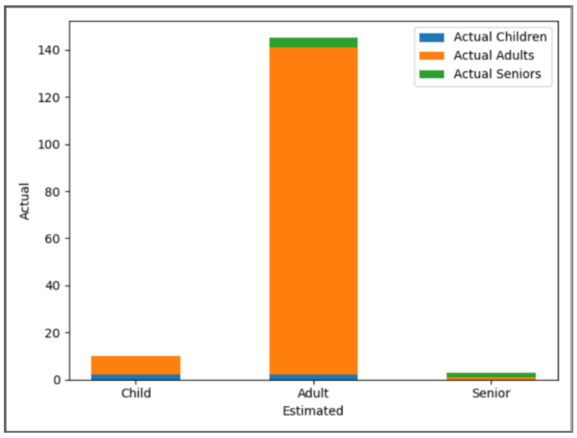


Figure 4.2: Age category accuracy (confidence threshold = 0.95)

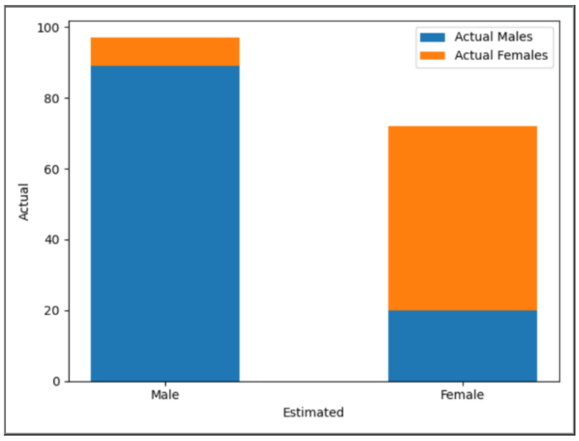


Figure 4.3: Gender accuracy (confidence threshold = 0.9)

While gender estimation seems to be very good for females and good for males, the estimations of age categories are a problem. As expected, a small percentage of adults get classified as seniors and a significant part of seniors gets classified as adults. The problem is that the number of seniors on Twitter (in this data set at least) is much less than anticipated and was stated in our research[26]. While we did acknowledge this data was from the United States and would thus probably not be exactly the same in the Netherlands/Flanders, we did not expect the number of seniors to be this low.

Our theory was that the small number of adults being classified as seniors would not influence senior sentiment a lot, since there would be enough seniors to counter these wrongly classified adults. Likewise, we expected the wrongly classified seniors to not influence adult sentiment a lot, since there are also enough adults to counter the wrong sentiment. Looking at the graph, it is visible that 50% of the classified seniors are actually adults and thus 50% of the senior sentiment is actually not from seniors. While at first glance this appears to be a problem, it is actually not that harmful. The adults that get classified as seniors were, in this case, people around 60 that looked old but were classified as adult. Having these people be classified as senior is not very detrimental to the senior sentiment, since these people are almost senior themselves.

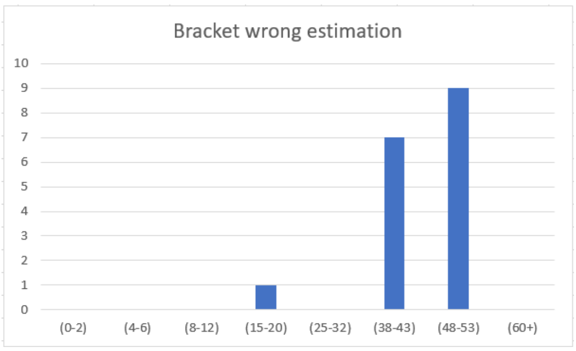


Figure 4.4: Age bracket accuracy (confidence threshold = 0.98)

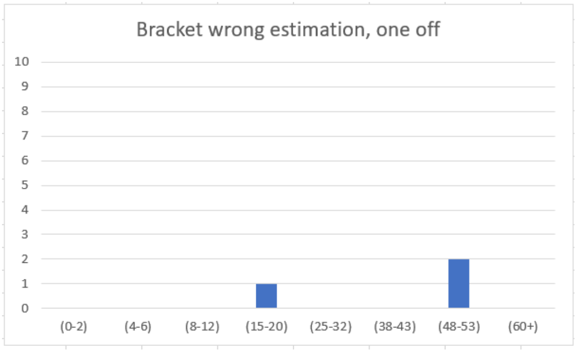


Figure 4.5: Age bracket accuracy, one-off error (confidence threshold = 0.98)

While this threshold is fairly high, it does give a high accuracy for one-off errors[2].

[1]We have made this data set ourselves, since there were no good data sets available. Since we inferred age and gender ourselves, there could be wrong estimations on our part during the annotation of twitter profiles

[2]While in the paper by Hassner and Levi, the age bracket (8-13) is used, the model actually returns (8-12), this is why in our

one-off errors are estimations that are one age bracket off. So for example, if the actual age is {4-6} but the estimation is {0-2}, then the estimation is one bracket off. If the estimation is {15-20} it is 2 brackets off, namely {8-13} and {15-20}.

4.4. Location filtering

To compare the sentiment of different locations, it is essential to be able to filter the tweets on a location. This was a bit of a challenge, as we discovered that of all the geotagged tweets, only a small number of them contained exact coordinates, in total too few to make an accurate analysis. However, all the geotagged tweets are associated with a ”place”, which is usually a city or point of interest (POI). When stream processing the tweets, each of them contains a bounding box of its associated place, so each tweet in our system is associated with a bounding box (see the red bounding box in figure 4.6. Besides that, it also contains a country code and place name.

To be able to search for locations, we used Nominatim's geocoder API. Geocoding is the process of turning a textual string into a place entity. This gives us information such as the type of the place (e.g. administrative, city, suburb, neighbourhood, etc.) and the polygon coordinates of the place, meaning the coordinates that make up the entire area of the place (see the green polygon in figure 4.6).

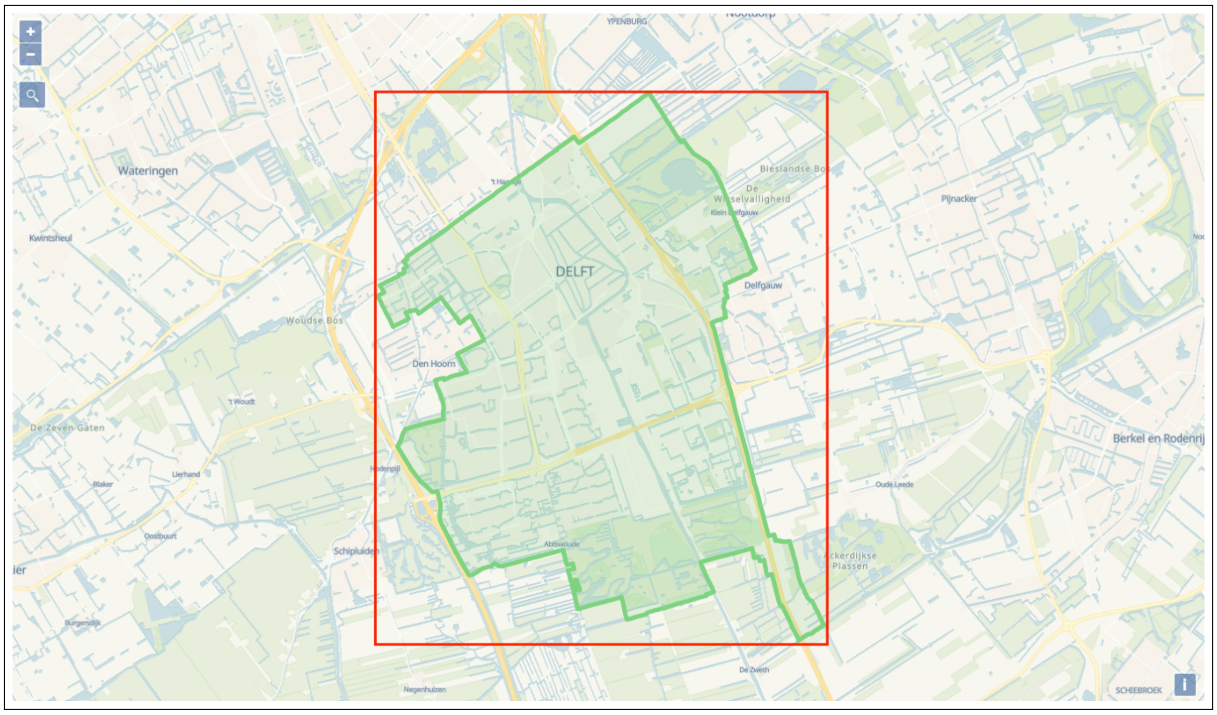


Figure 4.6: The polygon coordinates (green) and bounding box (red) of Delft displayed on the map.

First, we thought that comparing the polygon of a location with the bounding boxes of the tweets would give us the most accurate results. To achieve this, we would first geocode the query typed by the user and then intersect the place polygon with the tweet bounding boxes to see which tweets were posted within the boundaries of the searched location. However, we soon noticed that the places associated with tweets are often cities rather than neighbourhoods or POIs, meaning that the bounding boxes of these places were quite large and often included large areas that did not belong to the city. For example, the bounding box of the Hague overlaps with the polygon of Delft, thus all tweets from the Hague would be included if you searched for Delft. A solution for this would be to retrieve the exact polygon coordinates for every tweet in the database, but that would add significant overhead to the system (e.g. the polygons would need to be retrieved from somewhere and also intersecting with the exact polygons would take too much time for every single query). Instead, we solved this issue by matching locations on their name, which would also allow for significantly faster querying.

results (8-12) is mentioned instead of (8-13)

For the sake of clarity, we split up locations into three categories: countries, other administrative areas (cities, regions, provinces, etc.), and places within cities (neighbourhoods, POIs, etc.). For each of these categories, we decided to apply the name matching slightly differently to get the best results. How accurate these results are exactly is difficult to put into numbers, but we will discuss the major challenges encountered with each approach.

Firstly, searching for countries turned out to be the easiest, as both the tweets and the searched location have a country code that could be matched on. Matching country codes is easy because they conform to the same standard.

However, matching the other types of places came with an additional challenge, namely the fact that places can have ambiguous names. For these other types, we solved this by not only checking if the names match, but also checking if the bounding box of the tweet place and the polygon of the searched location intersect. When searching for a city, this gives quite a strong indication that the correct place is matched. In theory, it could still fail if a translation of the place name is missing, although we tried to solve that issue by persisting every encountered translation of the place names, which works quite well in practice. The approach still has some difficulties with more extraordinary cases, such as distinguishing the city and province Utrecht (the city is in a province with the same name). In fact, provinces in general are difficult to deal with using name matching, because when you search for a province none of the cities within the province will match the name, therefore only tweets tagged with the province as location will be included.

For the last category, places within cities, we applied the same approach of name matching and checking for intersection, but we included an additional check as well: a place will also be included if its bounding box falls completely within the polygon of the searched location. This has the advantage that when you search for a neighbourhood, POIs within that neighbourhood will also be included.

When searching for a city it would barely make a difference in sentiment to include neighbourhoods and POIs within cities, because the majority of tweets are tagged with a city as place. However, it would make the query significantly slower. Therefore, we decided not to do this for cities, as it would slow down the search dramatically without having a significant impact on the results. Contrary, in the case of searching for a neighbourhood it is useful to include the POIs inside of it, because they make up a large amount of the tweets posted within the area, therefore having a significant impact on the results. It also does not slow down the query as much, because the polygon for a neighbourhood is generally smaller than the polygon of a city and the complexity of the polygon is positively correlated with the time it takes to check the intersection.

When searching on the map in the web app, the polygon of the search result will also be displayed as in figure 4.6 to give some more insight into what is used to filter the locations of the tweets.

To summarise, when searching for a country all tweets associated with the corresponding country code will be matched. When searching for other types of administrative areas, all tweets with the same name that also intersect with the polygon of the location will be matched. And when searching for places within cities, all tweets that either match with name and intersection or fall completely within the polygon of the location will be matched. The approach is not 100% accurate but works well in practice.

5 Build and testing

This chapter describes the pipelines that were set up at the beginning of the project in order to actually run the code and confirm its working. It first briefly discusses how the project can be built and then further discusses how the code was tested.

5.1. Build

To make it easy to build the project from scratch, both the stream processing part and the web app part of the project contain their own build scripts. The first thing these scripts assure is that all the dependencies are installed. Since the project is written in Python and JavaScript, these dependencies are managed by pip and npm respectively and will be installed by these tools. For the web app, the JavaScript source code is then transpiled by Webpack to make sure that the browser can run it. After the JavaScript is transpiled, the Flask server will be started and the app is running.

To make sure that the code also meets the quality standards at all times, the build scripts also run the automated testing and static analysis tools, which are discussed in the next section.

5.2. Testing

To confirm the working of the product throughout development, automated testing and different static analysis tools were used.

For testing the back-end code (e.g. the stream processing and web app back-end), the project contains a Python unit test suite. For the build to succeed, all tests must pass and the total branch coverage must be at least 80%, although this number was typically throughout development in the 90%. A limitation of this test suite is that it currently does not include any system/integration tests. It would benefit from having some integration tests covering the integration with the external APIs that are used, as well as integration tests for verifying the integration of the stream processing and the web app.

To ensure the quality of the code, Flake8 and ESLint are used to lint the Python and JavaScript code respectively. The tools ensure that the code style always conforms to standards and that bad practices are avoided.

To easily confirm that all these checks pass successfully, all tests and static analysis tools are also run on a continuous integration server, meaning that all members of the team can see within the blink of an eye if the tests pass. This should prevent any unfinished code from being put in production.

6
SIG

At two moments over the course of the project, the entire codebase was evaluated by the Software Improvement Group (SIG). The feedback provided by them was taken into account to improve the code of the project. The first submission was made at the end of the fourth week of the programming phase, i.e. week 4.6 on the academic planning. Because we only received the feedback of this submission at the end of week 4.7, our codebase had already changed quite dramatically and a lot of the feedback was not applicable anymore. Moreover, there was only one week left to make improvements based on this feedback before the final submission at the end of week 4.8. It should be noted that at the end of week 4.8 this deadline was moved up one week by the BEP coordinators, as there was some confusion about the exact deadline.

6.1. First submission

Figure 6.1 shows the feedback we received from SIG after the first submission. Because the last three categories (e.g. unit complexity, unit interfacing, and module coupling) all received a rather high rating, we decided to focus mostly on improving duplication and unit size in our codebase.

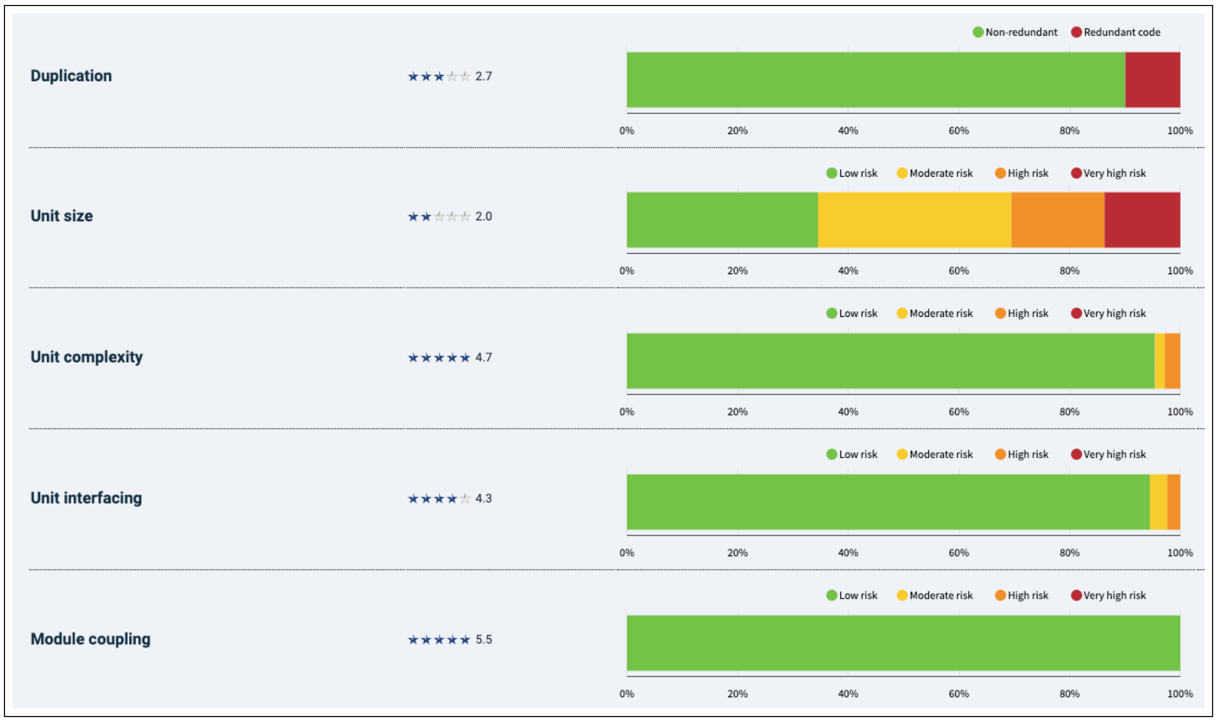


Figure 6.1: The metrics of the first SIG submission.

6.1.1. Duplication
Because our codebase had changed a lot already between making the submission and receiving the feedback, most of the duplication found by SIG was not applicable anymore. To still be able to make improvements in this area we used a tool called ”jscpd”, a copy/paste detector available through npm. We could run this tool ourselves from the command-line to see which files still contained duplication. The biggest issues we noticed were:

- React components with similar functionality;
- Redux reducers with similar functionality;
- reused functions for sentiment score/rating computation.

To solve the issue with react components, we looked into some React design patterns that could help us. Our first thought was to use inheritance to extract common functionality of the components in a superclass, but this soon turned out not to work very practical in JavaScript and against React practices according to its developers [27]. They recommend using composition instead. After extracting most of the similar functionality of the components using composition, the code duplication went down significantly.

As for the Redux reducers, we decided that it would be better to keep their structure unchanged, to stay in line with standard practices as much as possible. We valued compliance with standards more than the duplication metric.

To solve the last main issue that was caused by reused functions, we created a utility file containing all the functions that were reused. By doing this, we only needed to specify the functions once and could then simply import them wherever they were needed.

6.1.2. Unit size
To improve the unit size in the codebase we mainly focused on splitting up some larger methods. For example, we took a large set-up method in the file ”OpenLayersMap.js” and created a separate method for each part of the set-up. We also extracted some parts of a long method in the file ”age_gender.py” from the Twitterconsumer, moving this functionality to smaller methods with a more atomic responsibility each.

6.1.3. Unit interfacing
Although not a priority, some refactoring was done to improve unit interfacing. A few Python methods scored low on unit interfacing due to containing too many parameters. We refactored some of these, e.g. the methods in ”api.py” (now ”twitter_api_client.py”) were changed to a class. While reducing the method parameters, the new structure also made the code more readable. Another Python method with too many parameters (five) was ”create_tweet”. However, we decided to leave this method as is; reducing the number of parameters would also reduce the readability of the method since three or more parameters would need to be combined into a tuple.

6.2. Final submission
In the three weeks between the two SIG submissions, we improved the codebase according to the SIG feedback as described before and still implemented new features (e.g. wrote new code). Figure 6.2 shows the summary of the feedback we received from SIG after the final submission.

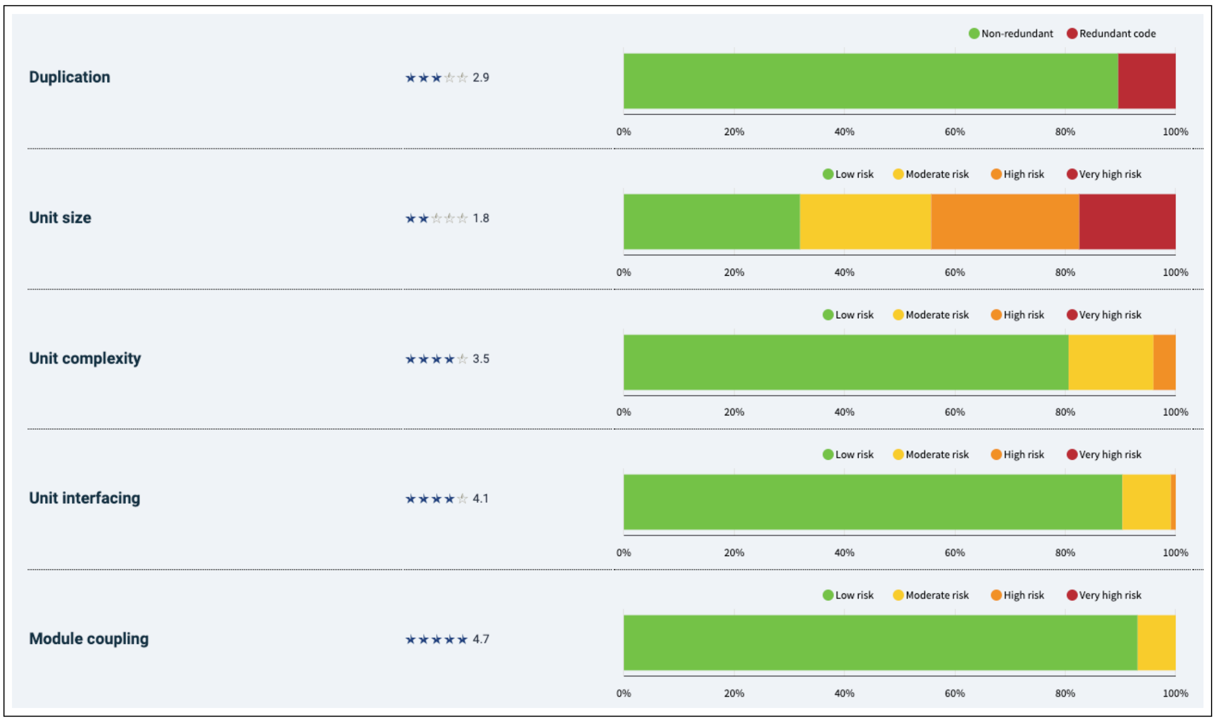


Figure 6.2: The metrics of the final SIG submission.

Looking back on the changes we made after the feedback on the first submission, we can see that the rating for duplication went up with 0.2, meaning our changes had a positive influence on the maintainability of the project as we expected. When taking a closer look at the violations that are still there, it seems that most of them have to do with the Redux code and a number of React components with similar properties. We discussed before why we decided not to change the Redux code and we think that React components with similar properties is something we can also do very little about, since each component makes use of the properties in different ways.

We were slightly surprised to see a decrease of 0.1 in the rating for unit size, although the reason for the decrease becomes clear rather quickly when looking at the violations: most of the violations have to do with long React render() methods. Since we added more components, it makes sense that we also have more of these methods. Although for some of them we tried splitting them up into smaller methods, for most of them that did not make much sense as it only made the methods more difficult to read. Therefore, we decided to keep most of these unchanged, despite a slight decrease in the unit size metric.

Unit complexity went down with 1.1, however, if we look at the actual violations SIG only shows us two, both of them in the JavaScript files. One of those seems to be due to SIG not properly handling React's JSX language and the other one is indeed a rightful violation where a method contains one if-statement that is handling too much logic. However, a rating of 3.5 while the only violation if this one if-statement seems to be more of a problem with the metric than with our codebase.

Unit interfacing went down with 0.3, when looking at the violations this seems to be due to one JavaScript method and three Python methods taking too many parameters (e.g. five or more, although in this case all of them take five). Since some of the code containing the violations was not yet submitted to SIG in the first submission, this was difficult for us to prevent. In retrospect, it would have been good to have a static analysis tool to check for this, although since the violations are rather small it might not make sense to change the code if it reduces readability.

Lastly, module coupling also went slightly down 0.8. However, SIG does not show us any concrete violations, so it is difficult to do anything with this point of feedback.

7 User test

To verify the user interface we built and find out about any flaws in user experience, we held a small user study. Keeping the COVID-19 measures in mind, we decided to do the user test through an online survey that we prepared beforehand instead of other forms of user testing like think-aloud tests. To make sure not to overload the server, since the server is currently configured to only handle a small set of users simultaneously, we only sent the survey to a small number of people per day.

The full survey we prepared can be found in appendix D. The survey included three small use cases in which the user is presented with a simple scenario and has to answer a question. This way we can check if the user was able to navigate the system well enough to answer a simple question correctly. In total, 21 people participated in the user study. All of these participants are students, which might have introduced a bias in the results.

To gain some insight into what the participants had difficulty with while using the application, we will go through the three cases one by one.

For case 1, 85.7% of the participants were able to get the correct answer. The additional remarks mostly contain small suggestions but there does not seem to be a certain cause for the incorrect answers.

For case 2, the percentage of correct answers was only 57.1%. In the additional remarks for this case, it stands out that six participants all noted that it would help to show the city names on the chart legend for the sentiment over time, rather than just displaying ”left” and ”right”. This is something that could be implemented to create a better user experience.

For case 3, 66.7% of participants answered correctly. An additional remark that seems odd is that one participant mentioned they had to think a bit about how to go about getting the answer, but then explained they had found the correct way to go about it (which was indeed the case). However, they still answered the question incorrectly.

Something interesting we saw after exploring the data is that even though the first case had the highest rate of correct answers, it was still considered most difficult by the users. On average on a scale of 1-5 (where 1 is very difficult and 5 is very easy), case 1 was got a score of 3.24 while case 2 and 3 both got a score of 3.33. This seems counter-intuitive, but we expect it is because when answering the first case participants were still familiarising themselves with the interface. After the first case, it felt easier to answer the other questions, even though fewer of them answered correctly.

Overall we found that some features of the tool can be unclear for new users. It was unclear for our users what the scale of the average sentiment score was, what the difference was between red and blue colors in the word cloud and why they could not search for data in May. Adding such information in the ’information page’ is not intuitive. Displaying such information when you hover over components would be a more useful addition for a better user experience.

8 Conclusion

Over the span of two months, we created a product that gives insight into the sentiment of tweets. The tweets are stream processed and analysed to extract the sentiment, various tweet characteristics (e.g. location, timestamp, etc.), and some user demographics (e.g. age and gender).

To establish the sentiment of a tweet, its text is first processed and then machine learning is used to determine the sentiment class. Machine learning is also used to estimate the age and gender of users based on their profile pictures. For location filtering, a geocoder API is used to match a search query to the locations of the tweets. Combining all this with an intuitive interface allows the user to easily do research on sentiment in the Netherlands and Flanders.

As of writing this report, the database of CITYSENT is relatively small (one month of data). But running the server will result in a bigger database over time. CITYSENT needs no external assistance and gathers sentiment data automatically. In a few months' time, the database will be significantly bigger. Our clients will then be able to see how certain events impacted sentiment and how people feel about specific places. With this information, they can try to influence future sentiment, through better city planning and thus contribute to a happier society.

9 Process evaluation and recommendation

9.1. Process evaluation

CITYSENT was made during very different circumstances than a normal Bachelor End Project. Because of the Covid-19 pandemic, having physical meetings was impossible and thus a process had to be adopted with which the team had little to no experience.

9.1.1. Three pillars

There were three points that were especially important during this project: regularity, availability and communication.

Regularity

Normally, regularity is not a priority for projects, since physical meetings automatically force teams to work on the project on a regular basis. Because this project had to be done through online meetings, it was very important to have a good work schedule. The team always met online at 09:00 to discuss what to do for the day. Then after an hour, the meeting ends and every member works on his or her assigned part. Then at the end of the day, another meeting took place, in which the goals for the day were evaluated and plans for the next days were made. Having two team meetings a day forced the group to keep up and work regularly.

Furthermore, the team had one meeting with the clients and one meeting with the coach every week. These meetings were used to reflect on the past work and plan the future features of CITYSENT. These meetings have proven to be very efficient in keeping a good overview of what the clients want, but also what the TU Delft expects of Bachelor End Projects.

Availability

Normally, since teams are working near each other, availability is no problem. Since our team had to work from home, we made the rule that everyone should be available on discord from 09:00 until 17:00. This was done to ensure that when questions had to be asked at the team, there was no time lost in waiting for a response. There was also a WhatsApp group for the team outside these hours and a WhatsApp group with the clients for mid weekly updates. WhatsApp was used because it allowed the team to quickly give small updates and also get fast responses back.

Communication

Communication is always an important aspect of project planning. For the team, it was important to clearly communicate with each other when things went wrong, so that other team members could jump in and help fix the issues. Still, fixing an issue with the use of pair programming when not physically near each other, is very hard to do. This is why we opted for CITYSENT to be a modular system. This allowed the team to work individually and easily combine different components with each other.

9.1.2. Scrum

The Scrum method was used exactly as stated in the project planning. Only in the last two weeks, the Scrum planning was less weekly and more daily based. This was done because the required time frame of certain implementations was not clear. Tackling said implementations on a daily basis (starting with implementations of highest priority), minimized the room for estimation errors and thus ensured the most important features were in place before the end of the deadline.

9.1.3. Issue board & merge request

The issue board and merge requests have been proven to be very important throughout the project. Both were very effective tools to update the team with and have discussions in, without actually having to start a meeting.

9.2. Recommendation

Because of the modular design of CITYSENT, new features are easy to implement. There are a few features we would have liked to be in CITYSENT, but due to time constraints, were not able to implement. These features are:

9.2.1. Sentiment analysis based on emoticons

A lot of tweets contain emoticons, which say a lot about the sentiment of a tweet. There are two ways to include emoticons:

- Train a machine learning model on a data set that has emoticons included
- Give emoticons a sentiment score and apply it to the score calculated by the machine learning algorithm that is running now

While the first might be easier to implement, since the only thing you need is a good data set, we haven't found a data set that satisfies our needs. Still, if a good data set is found the first one would be a great addition. If no data set can be found, the second option could be a sufficient alternative.

9.2.2. Language support outside of English and Dutch

For now, CITYSENT only supports Dutch and English tweets. It would be a good addition if other languages would also be supported. Since Google Translate is used to translate Dutch tweets to English, the same technique could be used for other languages.

9.2.3. Activity detection based on pictures contained in tweets

In the same way age and gender is inferred through profile pictures, it would be interesting to have a module that analyses images contained in a tweet and infer what activity is done in said image. This way, CITYSENT can show users which activities influence sentiment. For example, it might be that people in the center of The Hague are mostly happy when they are going out with friends at night, but when they are doing groceries during the day, sentiment might be lower.

9.2.4. Combination of third party data to calculate correlation

The initial idea was to include an option for users to import their own data and compare that to CITYSENT's collected data. Unfortunately, this idea was out of scope for the project and thus scrapped. We do believe that such a feature would be extremely valuable for researchers that have access to said data sets, like our clients from Urbanism have.

9.2.5. Support for other social media

We have noticed that a big chunk of Twitter users is 30-40 years old, which was expected. In order to get a better view of sentiment, other social media wherein other demographics are present should also be included. In our research, we concluded that using Instagram or Facebook was not desirable because of the costs to access said data. If, however, the clients think this is a worthwhile investment, this would be definitely something to look into.

9.2.6. Visually display sentiment in the map

Initially, we had planned to implement the map in such a way that the clients could see a clear overview of the sentiment distribution over the different locations, for instance by color differentiation. Unfortunately, this turned out to be too much work for the scope of our project. Since we did implement the top cities and lowest cities charts, we decided it was okay to leave the map as it is right now.

9.2.7. Amount of users

During the project peculiar data passed by. For example, at one point we found out that more than half of the negative tweets we streamed in were posted from Menaldumadeel, which is a little town in Friesland. Searched for Menaldumadeel on Twitter, we found out that this was largely due to a weather station. We therefore decided to filter out all tweets from accounts that do not have a face in their profile picture. This way you exclude for instance companies, weather stations and news channels. Unfortunately, this does not filter out all peculiar data. For instance, we found that there was one woman in Texel who tweeted a lot, and therefore influenced the data about Texel a lot. For this reason, we think a nice enhancement to our tool would be to include information about the amount of users the data is based on. This would, however, make the stored data less anonymous since the system would need to be able to know which tweets were posted by which users. Sufficient security measures would need to be implemented to ensure that data cannot be traced back to individual users to protect users' privacy in the case of a data leak.

9.2.8. Widen target audience and authentication

Currently, CITYSENT is only accessible on the TU Delft networks and therefore targets only people with access to these networks. This was a deliberate decision made together with the clients to restrict the access to the sentiment data set used by CITYSENT. In the future, CITYSENT could be made publicly accessible to anyone in the world while at the same time restricting data access to trusted users. This would require a simple change: changing the firewall rules to allow traffic on ports 80 and 443 from anywhere instead of TU Delft-only. Before allowing access to anyone, a log-in/authentication process should be implemented to restrict and control access to the downloading of the application's data set. Although the data is anonymous, restricting download access to authorized users would prevent unauthorized users using CITYSENT as a means of collecting Twitter data (without applying for API access), which is not the purpose of CITYSENT.

9.2.9. Filtering on keywords

CITYSENT already extracts keywords from tweets and provides the user the ability to see which keywords are related to positive, negative and neutral sentiment in the dashboard. To make this feature even more useful, the possibility to filter on keywords could be added. This would allow sentiment analysis on specific topics represented by keywords.

10 Ethical implications

10.1. Improved liveability

The main ethical implication of CITYSENT is its contribution to the design of better cities and thus the improvement of liveability. CITYSENT will allow city planners to see which factors make people happy and plan accordingly in future projects. This will likely increase the happiness of the people that are affected by these projects, which is ethically a good thing to contribute to.

10.2. Privacy

Since CITYSENT processes user-generated content, it is crucial to harbour the privacy of said users. This is done by not saving the content or profile of the author of a tweet. Only the sentiment, location, age and gender will be stored.

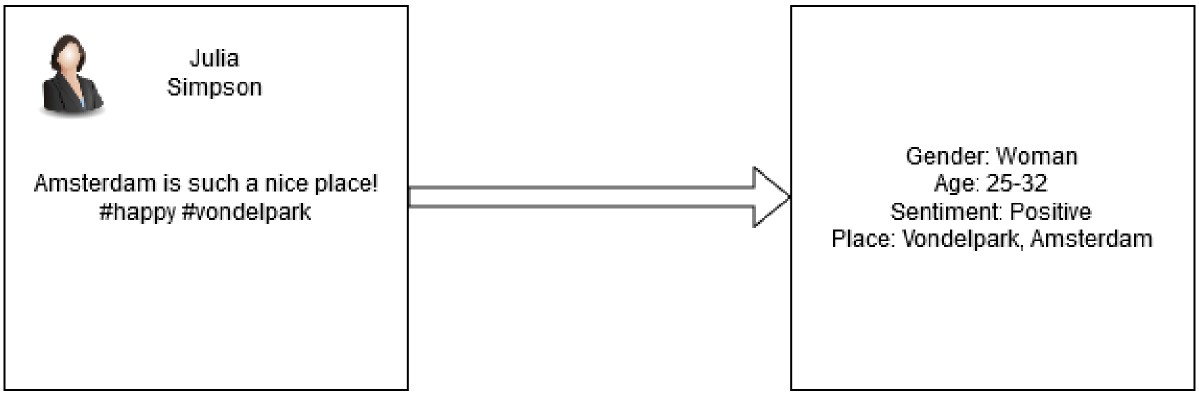


Figure 10.1: Tweet to processed data

Processing tweets immediately and not storing personal information, ensures the privacy of the author is secured. It is impossible to deduce who the author is, which is not only ethically responsible but also in compliance with GDPR [28].

10.3. Data accessibility

Throughout the research phase, it became extremely clear how much power social media platforms actually have and how valuable their data is. Most big platforms keep their data behind a paywall, which makes it inaccessible for many people. Since CITYSENT is also collecting and processing data, it is important to be critical of the position we are in. On one hand, having the data accessible to only the clients will give them an advantage over other city planners that do not have access to said data. On the other hand, sharing CITYSENT's data with others might lead to more researchers gaining better insight into the sentiment of the Netherlands and Flanders. These researchers can then use this information to better their own research.

Whether the data we have gathered will be shared with third parties, will be up to the clients.

10.4. Generalisation & false implication

Like with all data-driven tools, it is often easy to generalise. If, for example, our tool shows that mainly 23-year-old males are unhappy near big malls, does not mean every 23-year-old male is. Users of CITYSENT must be aware that unintended generalisation could happen and influence their research negatively.

When comparing data sets it is important to remember the famous statistics phrase: ”Correlation does not imply causation”. An example of this is sentiment at a park. A researcher might see that the more parks are present in a city, the happier its residents appear to be. This could mean that parks make people happier, but it could also be that rich cities have a bigger budget for more parks. Rich cities also have a bigger budget for other things that increase liveability and thus are more likely to have happier residents. If the latter is the case, people are not getting happier because of parks, but because they live in more prosperous cities. The extra parks are just a side effect of a city being prosperous.

Bibliography

[1] J. Isaak and M. J. Hanna, “User data privacy: Facebook, cambridge analytica, and privacy protection”, *Computer*, vol. 51, no. 8, pp. 56–59, 2018.

[2] J. Perriam, A. Birkbak, and A. Freeman, “Digital methods in a post-api environment”, *International Journal of Social Research Methodology*, vol. 23, no. 3, pp. 277–290, 2020.

[3] M. Zimmer, ““but the data is already public”: On the ethics of research in facebook”, *Ethics and information technology*, vol. 12, no. 4, pp. 313–325, 2010.

[4] E. Kernel, *Three biggest legal cases about data scraping*, Dec. 2019. [Online]. Available: https://jaxenter.com/data-scraping-cases-165385.html, (Accessed on 30/04/2020).

[5] Twitter, *Apis and tools to tap into what’s happening*, 2020. [Online]. Available: https://developer.twitter.com/en/products/products-overview, (Accessed on 16/06/2020).

[6] A. Kafka, *Introduction*, 2017. [Online]. Available: https://kafka.apache.org/intro, (Accessed on 16/06/2020).

[7] L. Johansson, *Part 1: Apache kafka for beginners - what is apache kafka?*, Mar. 2020. [Online]. Available: https://www.cloudkarafka.com/blog/2016-11-30-part1-kafka-for-beginners-what-is-apache-kafka.html, (Accessed on 16/06/2020).

[8] E. Solovey, *Handling five billion sessions a day – in real time*, Feb. 2015. [Online]. Available: https://blog.twitter.com/engineering/en_us/a/2015/handling-five-billion-sessions-a-day-in-real-time.html, (Accessed on 16/06/2020).

[9] N. T. Blog, *Evolution of the netflix data pipeline*, Feb. 2016. [Online]. Available: https://netflixtechblog.com/evolution-of-the-netflix-data-pipeline-da246ca36905, (Accessed on 16/06/2020).

[10] I. Diffy and N. Hanzlikova, *Running kafka streams applications in aws*, Nov. 2017. [Online]. Available: https://jobs.zalando.com/en/tech/blog/running-kafka-streams-applications-aws/, (Accessed on 16/06/2020).

[11] E. Rescorla, “Http over tls”, RFC Editor, RFC 2818, May 2000. [Online]. Available: https://tools.ietf.org/html/rfc2818.

[12] *Free and unlimited google translate api for python¶*. [Online]. Available: https://py-googletrans.readthedocs.io/en/latest/.

[13] [Online]. Available: https://translate.google.com/.

[14] O. Kolchyna, T. T. Souza, P. Treleaven, and T. Aste, “Twitter sentiment analysis: Lexicon method, machine learning method and their combination”, *arXiv preprint arXiv:1507.00955*, 2015.

[15] A. Giachanou and F. Crestani, “Like it or not: A survey of twitter sentiment analysis methods”, *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–41, 2016.

[16] T. Mitchell, “Machine learning, mcgraw-hill higher education”, *New York*, 1997.

[17] V. Narayanan, I. Arora, and A. Bhatia, “Fast and accurate sentiment classification using an enhanced naive bayes model”, in *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, 2013, pp. 194–201.

[18] C. Cortes and V. Vapnik, “Support-vector networks”, *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[19] *Learn*. [Online]. Available: https://scikit-learn.org/stable/.

[20] *A twitter sentiment analysis tool*. [Online]. Available: http://www.sentiment140.com/.

[21] Sumit Saha, *A comprehensive guide to convolutional neural networks — the eli5 way*, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, (Accessed on 04/06/2020).

[22] Gil Levi and Tal Hassner, *Age and gender classification using convolutional neural networks*, https://talhassner.github.io/home/projects/cnn_agegender/CVPR2015_CNN_AgeGenderEstimation.pdf, (Accessed on 04/06/2020).

[23] spmallick, *Code for age gender recognition using deep learning*, https://github.com/spmallick/learnopencv/tree/master/AgeGender, (Accessed on 24/06/2020).

[24] M. Aiken and Z. Wong, "An updated evaluation of google translate accuracy", *Studies in linguistics and literature*, vol. 3, no. 3, pp. 253–260, 2019.

[25] H. M. Pan, *How bleu measures translation and why it matters*, Nov. 2016. [Online]. Available: https://slator.com/technology/how-bleu-measures-translation-and-why-it-matters/.

[26] J. Clement, *Percentage of u.s. adults who use twitter as of february 2019, by age group*, https://www.statista.com/statistics/265647/share-of-us-internet-users-who-use-twitter-by-age-group/, (Accessed on 29/04/2020).

[27] React, *Composition vs inheritance – react*, https://reactjs.org/docs/composition-vs-inheritance.html, (Accessed on 24/06/2020), 2020.

[28] The European Union, *What is considered personal data under the eu gdpr?*, https://gdpr.eu/eu-gdpr-personal-data/, (Accessed on 30/04/2020).

A
Project info sheet

Info sheet CITYSENT
TU Delft: *Department of Urbanism*
Presentation to be given on 1 July 2020

CITYSENT extracts and analyses live Twitter data and stores the results in a database. For every tweet, sentiment is inferred based on the content of said tweet. Through the profile picture of the tweet author, age and gender is inferred and, based on geolocation, the place where the tweet originated from is stored. Displaying all this information in an easy accessible way, allows researchers to effortlessly see trends regarding sentiment in Twitter data. These trends can then be filtered on age, gender and place.

The main thing we learned is how powerful big data companies (such as Facebook and Twitter) are. While the users give all their data to them for free, accessing it is often put behind a paywall. Only Twitter allows access to a stream of recent tweets, without payment. This is why CITYSENT focusses on Twitter data, instead of more mainstream platforms such as Facebook or Instagram.

Because of COVID-19, all work had to be done from home. We opted for CITYSENT to be a modular system, so that it is easy to add new features, without being physically together.
CITYSENT will be used by our client to see what causes certain sentiment and if said causes can be replicated in future projects, so that sentiment can be positively impacted.

Team

Please note that the undermentioned roles are not the only things every member did, but merely the features he or she spent most of their time on.

Daan Goslinga
Interests: artificial intelligence, machine learning & big data
Roles: age/gender estimation, user interface, tweet translation, report

Raoul Kalisvaart
Interests: machine learning, big-data, data science, application development
Roles: preprocessing, tweet translation, sentiment analysis, keyword-generation, front-end, API integration, report

Adriaan Pardoel
Interests: machine learning, big data, GIS
Roles: project set-up, map, tweet filtering, CITYSENT API, front-end, report

Cathrine Paulsen
Interests: big data, stream processing
Roles: stream processing, database setup, server setup and deployment, download feature, front-end filters, report

Sarah de Wolf
Interests: user interface and machine learning
Roles: user interface, report

Project Coach: *G. Migut* *TU Delft EEMCS*
Project Clients: *Dr. E. J. Meijers & Dr. C. E. L. Newton* *TU Delft Urbanism*
Contact: *Daan@Goslinga.nl, Raoul.kalisvaart@planet.nl*

The final report for this project can be found at: http://repository.tudelft.nl

B
Original project description

The project aims to advance our understanding of people's feelings towards cities. People express these feelings using pictures, words emoji's, etc. The use of social media, and the sharing of images and descriptions about certain city spaces on these platforms is an expression of these feelings.

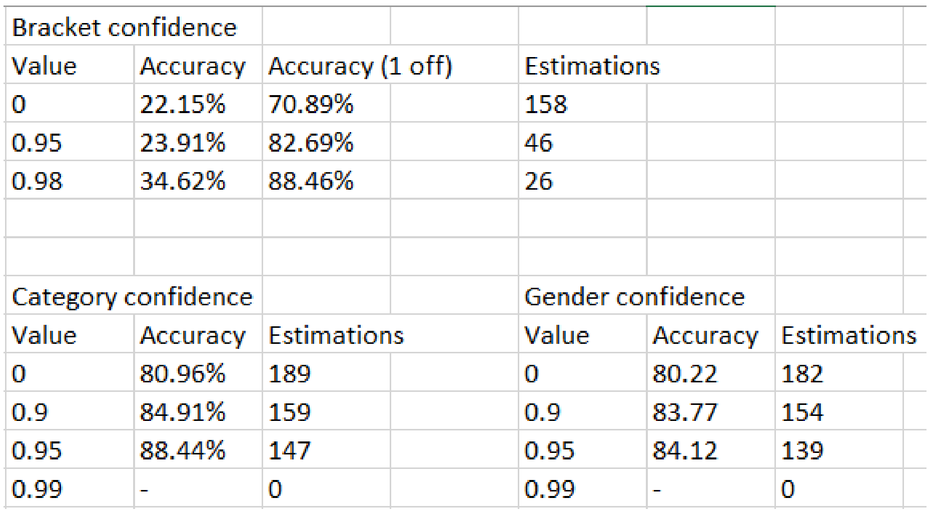
In a practical sense we want to analyse the sentiments that different people have regarding a set of pre-defined cities. To get an insight into these sentiments we want to analyse the appreciation of specific cities on different social media, such as Instagram, facebook, twitter, and others. Which get positive reports, and which get a 'bad rap'? E.g. are people more positive about Amsterdam compared to Rotterdam for instance? Or do sentiments vary between different areas/neighbourhoods of such cities? The first level of analysis looks at the emotional expressions people voice (in words or using emoji's) when they are in certain cities or particular parts of these cities. A second level of analysis includes the visual analysis of the images that accompany the messages, the emoji's or hashtags. The aim is to see if there is a relation between certain images of specific places and the sentiments (that can be extracted from the accompanying words, emoji's or hashtags) that people have of these places or cities. Additionally we aim to understand if there are different sentiments for people with different attributes (eg. Tourists versus inhabitants, or women versus men, urbanite versus villager, etc…).

We expect that we will be able to identify clusters of words and images that make emotional evaluations of diverse cities and intra-urban situations. Perhaps there are existing lexicons already that can be used. The analysis should allow a clustering of the spatial settings with the sentiments. The exact definition of the project will be done in consultation with the student team + mentor.

Students will apply:

- Named entity recognition for city/neighbourhood names
- Text mining (Emoji mining?)
- Machine learning related to images + to identify characteristics of people
- Sentiment analysis
- Map-making
- Database creation (the final product) in which a range of cities (+neighbourhoods) is associated with sentiments of different groups in society (gender, age, tourist/local etc.).

C Confidence thresholds



Bracket confidence			
Value	Accuracy	Accuracy (1 off)	Estimations
0	22.15%	70.89%	158
0.95	23.91%	82.69%	46
0.98	34.62%	88.46%	26

Category confidence				Gender confidence		
Value	Accuracy	Estimations		Value	Accuracy	Estimations
0	80.96%	189		0	80.22	182
0.9	84.91%	159		0.9	83.77	154
0.95	88.44%	147		0.95	84.12	139
0.99	-	0		0.99	-	0

D
User test survey

CitySent

CitySent is a tool developed for the Bachelor End Project of the Delft University of Technology. It can be used to research sentiment in cities across cities in the Netherlands and Flanders. Through the processing of Tweets, sentiment data is extracted and put in a database. This database can be interacted with through a user interface (a website). This survey aims to collect opinions on the CitySent interface so that we can see which parts still need improvement to be more intuitive.

In this survey, you are asked to perform three small tasks and voice your opinion of the platform. There are no time limits. You are not forced to answer any of the questions. If you do not want to answer a question, you can leave it blank. English answers are preferred but Dutch is also fine. The data you provide is completely anonymous, will not be shared with any third parties, and will be used for purposes of this research only.

On your computer, please go to: https://citysent.ewi.tudelft.nl

Take a brief moment to familiarize yourself with the look of the platform.

What is your profession/field of study?

Jouw antwoord

On first impression, what would you expect to be able to do with CitySent?

Jouw antwoord

On a scale of 1-5, how clear is the purpose of the "Overview" page to you?

	1	2	3	4	5	
Very unclear	○	○	○	○	○	Very clear

On a scale of 1-5, how clear is the purpose of the “Analysis” page to you?

	1	2	3	4	5	
Very unclear	○	○	○	○	○	Very clear

Volgende

CitySent

Use cases

Use Case 1

Between 10-06-2020 and 14-06-2020, on average, did women display a more positive sentiment in Amsterdam or in Antwerp?

Amsterdam or Antwerp? (use case 1)

○ Amsterdam

○ Antwerp

How easy was it to find this answer? (use case 1)

	1	2	3	4	5	
Very Hard	○	○	○	○	○	Super Easy

If you have any additional remarks, please leave them here (use case 1)

Jouw antwoord

Use Case 2

Between 10-06-2020 and 14-06-2020, on which day(s) did women in Amsterdam display a more positive sentiment than women in Antwerp?

Which days? (use case 2)

- ☐ 10-06-2020
- ☐ 11-06-2020
- ☐ 12-06-2020
- ☐ 13-06-2020
- ☐ 14-06-2020

How easy was it to find this answer? (use case 2)

	1	2	3	4	5	
Very Hard	○	○	○	○	○	Super Easy

If you have any additional remarks, please leave them here (use case 2)

Jouw antwoord

Use Case 3

On 13-06-2020, on average, did women or men display a more negative sentiment in Rotterdam?

Women or men? (use case 3)

- ○ Women
- ○ Men

How easy was it to find this answer? (use case 3)

	1	2	3	4	5	
Very Hard	○	○	○	○	○	Super Easy

If you have any additional remarks, please leave them here (use case 3)

Jouw antwoord

Vorige Volgende

CitySent

Concluding questions

Which features are, in your opinion, the most useful?

- ☐ Average sentiment
- ☐ Number of Tweets
- ☐ Sentiment score over time
- ☐ Number of tweets per sentiment over time
- ☐ Distribution of gender
- ☐ Distribution of age
- ☐ Most used keywords
- ☐ Highest sentiment cities
- ☐ Lowest sentiment cities

Which features are, in your opinion, the least useful?

- ☐ Average sentiment
- ☐ Number of Tweets
- ☐ Sentiment score over time
- ☐ Number of tweets per sentiment over time
- ☐ Distribution of gender
- ☐ Distribution of age
- ☐ Most used keywords
- ☐ Highest sentiment cities
- ☐ Lowest sentiment cities

Do you have any final remarks?

Jouw antwoord

Vorige Verzenden

E

Research report

Research Report

May 2020

1 Introduction

Through the immense stream of data that is uploaded to social media every day, it is often hard to keep track of what the current sentiment about certain places is. Knowing what certain people think or feel about specific places is very useful for future projects, such as city planning. Elements that appear to frequently cause an increase in positive sentiment might be something that entrepreneurs want to add to their businesses. Likewise, elements that cause negative sentiment, might be something they want to avoid. Having a tool that shows how sentiment changes over time and which variables have an effect on said sentiment, can be very valuable. City planners might, for example, want to know whether trees cause people to be happier on average, so that they might consider creating greener cities. Restaurant owners might want to know which age group thinks positively about the area they operate in and adapt their restaurant accordingly. The list of possibilities goes on and on, but it is clear that a tool that makes sense of all the data on social media is very valuable.

The goal of this research report is to find out how such a tool can be created, in a scope that is realistic for a Bachelor End Project.

2 Problem definition and analyses

2.1 Problem definition

As mentioned above, the biggest challenge is to make sense of the immense stream of data that is available to the public. If someone posts from a certain place, it does not mean the post and sentiment of the person is about said place. Vice versa, someone from the Hague could post something about Amsterdam. The sentiment should then be classified into Amsterdam and not the Hague. Some posts might not even contain sentiment at all. Think for example of political parties or companies posting ads on social media.

Also, getting to the data in the first place has been made increasingly difficult by companies such as Facebook and Twitter. In Facebook's case, getting the data without paying violates their terms of service and thus impossible without spending money. Twitter has a free, but limited, API and allows scraping to some extent. Although not ideal, the alternative is paying for access to data.

Lastly, next to acquiring and organizing the data, displaying it in an easily readable way is also a challenge. Since our tool deals with massive amounts of data, creating something readable might prove to be very hard.

A few questions the client likes to have answered are:

- Does COVID-19 lead to a more negative sentiment about cities, whereas the sentiment about rural regions has become more positive over time?
- With what attributes are positive and, respectively, negative sentiments about cities as expressed in words or in emoji's in tweets associated? (e.g. food, events, cultural activity, social gatherings, heritage, crowdedness, traffic, ...)
- How are these positive and negative sentiments spread over neighbourhoods within large metropolitan cities?
- To what extent is there a difference in evaluation of cities (sentiments) between different groups (tourists versus locals, male versus female, young versus old)?

Solving the aforementioned problems allows us to create a tool that the client can use to answer these questions.

2.2 Current situation

Before starting the project, it is always good practice to investigate whether other tools already provide the service you need. What we found was that there is already a tool that focuses on sentiment analysis of social media data: Zencity. However, this tool focuses on city planning, while CITYSENT's focus is more on general research. Because of this, CITYSENT will not provide the exact same service as Zencity and is thus still a valuable project.

Another tool that is very similar to our project is Social Glass. This tool was made by researchers at the TU Delft and "contributes new methods and tools for human-enhanced social data acquisition, integration, enrichment, analysis, and visualization". We reached out to the principal investigator of this project, Dr. Alessandro Bozzon, to get a better overview of their project and how we can fill in valuable gaps that they are not focusing on.

In short, Social Glass is not running their content analysis modules anymore, since they are pursuing different use cases. Looking at their website, it seems that crowd management is one of those use cases, which we are not interested in. The goal of CITYSENT is to get a good overview of sentiments of certain cities/areas and see how certain variables influence such sentiments. Content analysis plays a major role in determining sentiment and since Social Glass is not doing this anymore, it would be valuable to fill in this gap with CITYSENT.

The research done by the team of Social Glass could prove to be very useful to us. Their team has written papers about data extraction and architecture, which Dr. Alessandro Bozzon sent to us. A few of these papers also appear in this report. The goal of CITYSENT is not to reinvent the wheel, but to build further upon research done by others. As such, contact with Dr. Alessandro Bozzon is valuable to us and might be useful in further stages of the project.

2.3 Our assignment

Our assignment is to create a tool that can gather tweets from Twitter and process the sentiment contained in said tweets about certain locations. Users must be able to search for a place in the database and see the sentiment associated with it. The tool should be able to display sentiment scores over time, so that users can compare changes in sentiment. The client finds it interesting to compare the sentiment of different kind of people, thus the tool should classify tweets into groups (such as: age, gender and living area). Age and gender should be deduced through pictures, since getting that data is often not possible through APIs or scraping. Since the data should be easily readable, it should be displayed on a map. These are the most important functions of the tool. Other functions such as support of other languages than Dutch or English will only be implemented when there is time left for them. For a more detailed overview, look at the MoSCoW list in the project planning.

3 General structure

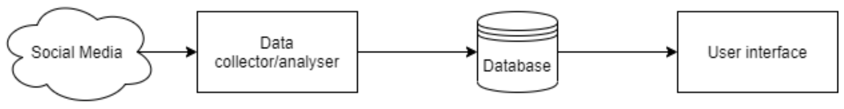


Figure 1: A general overview of entire system

The system will consist of two main components. The first of these is the part of the system that will handle the collection and sentiment analysis of the data. The other component will be responsible for presenting the analyzed data to the user in the form of an interactive interface. Both of these components are connected through a database. An overview of this can be found in figure 1. As the data collection and analysis component is relatively complicated by itself, the remainder of this chapter discusses that component in greater detail.

3.1 Data processing

An overview of the structure of the system's data processing subsystem, can be seen in figure 2.

The first component of this subsystem is the **data collection** component. This component is responsible for retrieving data about cities from social media platforms.

After the collection is done, the data is provided to the **data separation** component. This component is responsible for splitting the obtained social media data into several parts: a textual part (containing the textual information of a social media message), a geolocation and a profile picture.

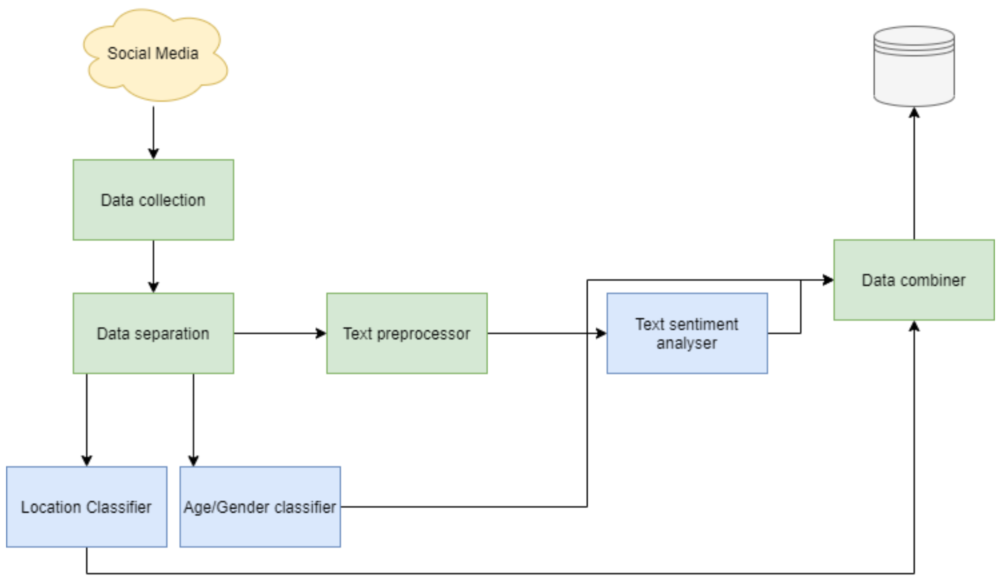


Figure 2: A general overview of the data processing component of the system

The textual data of the data is provided to the **text preprocessor** component, which performs several preprocessing operations on the data in preparation for the analysis of this data. The analysis is performed by the **text sentiment analyzer** component. This component uses the preprocessed textual data to infer the sentiment of a social media message.

Lastly, the **age/gender classifier** component uses potential profile pictures contained in the social media messages to infer the age and gender of the author of the message and the **location classifier** component uses potential geolocation data from the social media posts to infer a location.

After processing, these different pieces of information obtained by analyzing the different data sources need to be combined. This is done by the **data combiner** component, which also prepares the data to be stored in the database.

In the remainder of this report, the different components presented in this chapter are explained in detail.

4 Obtaining data

This section will describe the options for which social media platforms to retrieve data from and how, as well as how data should be properly handled in terms of users' privacy.

4.1 Data retrieval

A small but nonetheless important part of our system is the method of data retrieval, which falls under the data collection component. The main method of data retrieval from web-based services like social media platforms is through their own *application programming interface* (API). The API allows access to retrieve certain data contained on the platform in a semi-structured format (such as JSON). Retrieving data through an API makes the data retrieval easier. Since the retrieved data is already in a semi-structured format, less preprocessing is needed to proceed with further analysis. However, access to APIs has become increasingly more restricted in recent years in light of privacy concerns and the Cambridge Analytica scandal [1]. Subsequently, all large social media platforms (Facebook, Instagram, Twitter) have shut down their public APIs. Access to their APIs requires submitting a detailed application to the platform which they need to approve.

In the wake of the new API restrictions, researchers have taken to web scraping to obtain data for research, a topic of active legal and ethical debate [2]–[4]. Web scraping automates the process of manually collecting public data, effectively imitating a user browsing the site and downloading any relevant information. The downloaded information will be unstructured and therefore needs to be parsed. The process of web scraping is complex as it requires manipulation of http requests [5]. Although scrapers already exist for well known social media platforms (such as Instagram Scraper and Twitterscraper), even the smallest updates to the website and its HTTP framework can completely break the scraper, rendering it useless.

Therefore, using an API to retrieve data is the most robust and future-proof option, as well as the most convenient for further processing. Out of the big social media platforms, Twitter has the least restrictive API and it provides valuable information like geolocation that otherwise would not be available through scraping. The only downside is that the (free standard) API can only access tweets from the past seven days. However, we consider this is as a minor inconvenience as a data set of past tweets can be built over time so temporal comparison is still possible. The different sides to using APIs and scrapers for data retrieval will heavily influence our analysis of the viability of social media platforms.

4.2 Data sources

Firstly, it is important to consider which social media platforms are most suitable to use as data sources for the application. If we take a look at the Social Media Survey 2020 in the Netherlands held by Newcom Research [6], we see that the social media platforms with the most daily users (in descending order) are Facebook, Instagram, and Snapchat. Of these, Facebook and Instagram seem the most useful for the purpose of the application, as they deal with a large archive of data of which a good part is publically available, while Snapchat mostly holds onto data from the last 24 hours and is more private.

When looking further into the options regarding data from Facebook and Instagram, it should be noted that both platforms are owned by Facebook. They both used to have their own APIs through which a lot of public data was available. However, as previously discussed, around 2018 access got a lot more restricted [7], [8], making it impossible to gather large amounts of data through these platforms. At this time their APIs do not have options to carry out searches, which means we cannot retrieve posts belonging to specific locations for the purpose of the application. Moreover, it is possible to carry out searches through a web browser, raising the question of whether it would be possible to retrieve the data by scraping the web page. However, Facebook and Instagram officially disallow scraping without their written permission, therefore it would be highly unethical to do this and reaching out to them to get written permission seems like an impossible task, as Facebook seems to be very unreachable.

This brings us to the alternatives: other widely used social media platforms found in the survey are Twitter, TikTok, Tumblr, YouTube, Pinterest, and LinkedIn (again in descending order). We will discuss Twitter more in-depth as it turns out to be the most interesting platform to look at, because the other ones mentioned each have their own critical disadvantages: TikTok is mostly used by very young girls [6] and is more about sharing funny videos rather than expressing opinions about places. Tumblr also has a young audience (though not as young as TikTok) which is predominantly female [9]. YouTube has more diverse users [9] and has some useful opinions in the comments, but the comments seem very biased by the topic of the video. For example, we found that searching for "Leiden" on YouTube gives us videos about trips to the city with mostly positive comments from tourists and locals, while on the other hand it also gives us videos about students in Leiden with mostly negative comments from people expressing a strong dislike of students. Pinterest also has a mostly female user base [9] and the content on there is often collections of photos and links to other sites, rather than people expressing a sentiment. LinkedIn has the very specific purpose of maintaining a professional network, also making it a platform where not many opinions about cities are shared.

Twitter is an interesting case. Their demographics are fairly well distributed [9] and the platform has many daily users [6]. Although a lot of posts on there seem to be companies or organizations sharing links, there is also a good portion of people expressing opinions on there which would be useful for the application. Twitter even has an extensive API, but making use of it is quite expensive and the free version only lets you query tweets from the last seven days. However, they allow scraping to some extent, making it possible to use it as a data source for the application. With Twitter being one of the only accessible sources that has a large, diverse audience and useful content, we decided to use this as the main source the application will rely on. Some key issues with using Twitter are that profiles do not include demographics (age, gender) and tweets are often not geotagged (less than 1% of the tweets is geotagged). Ways to handle these challenges are more thoroughly discussed in section 6.

Because the useful data available from Twitter will still be limited, it might also be helpful to use data from sources such as TripAdvisor, although it should be noted that TripAdvisor specifically does not have a public API and does not allow scraping their website. While not strictly being social media platforms, these kinds of sources immediately give us more insight into how cities and places within cities are regarded by tourists. One platform in specific that seems interesting is Foursquare, because it is in-between a social media platform and a place review platform.

Another limitation of Twitter is that not all posts contain images like on a platform such as Instagram. Yet images could be useful to do analysis on for example what type of activities are popular in certain places. Therefore we considered if there would be any viable alternative for Instagram. The best thing we could find is Flickr, which used to be more of a social photo sharing website similar to Instagram, but is now mostly dominated by amateur/professional photography. Virtually all of the comments have a positive sentiment and most of them are about the photographs rather than the places where they are taken, rendering them quite useless for the purpose of the application. Therefore, this does not seem a valid replacement for Instagram and will not add much value to the application.

4.3 Privacy

Although the main use case of the final application is to be used internally, it is still important to make sure that privacy is properly protected in case of a data leak. Even if we get permission to use Twitter's API to retrieve data, we still need to handle the data in compliance with Twitter's terms of service (ToS) as well as the GDPR. To this end, we need to handle the collection and storage of personally identifiable information (PII) with care [10]. Coincidentally, Twitter does not store nor provide access to most PII through their API. However, we will be analyzing the geolocation of tweets and potentially the (general) location of users to be able to reason about spatial factors that could affect sentiment. According to the European Union [11], location data counts as PII and therefore falls under GDPR [11].

What this means in practice is that we need to anonymize the data such that no data point can be traced back to an individual user since we cannot realistically acquire consent from every user to store PII. Moreover, to comply with the Twitter ToS, if a tweet is deleted or modified we would also need to incorporate those changes if we store the tweet, which is practically impossible for the scope of this project. However, since we are only interested in aggregated statistics in the end, there is no real necessity to store specific tweets regardless. Privacy issues concerning PII are therefore solved naturally by the fact that we will only store anonymous and aggregated data. For instance, we would only store the fact that someone in Amsterdam tweeted positively about the area rather than the fact that a specific user posted this exact tweet at this exact location.

5 Sentiment analysis

The main component of CITYSENT is the sentiment analyzer. This analyzer can be used to process the data to determine whether collected tweets are positive, negative or neutral. The analysis of these tweets can roughly be divided into two parts: preprocessing the tweets and the sentiment analysis itself. In this section, both elements are explained in detail.

5.1 Preprocessing

In the majority of data mining applications, preprocessing input data is a vital step in the data analysis cycle. Preprocessing namely enhances the quality of the data and data of higher quality often results in the discovery of more useful patterns in the data [12].

In the field of sentiment analysis on social media messages, this statement also holds. Tweets posted on Twitter, for example, often contain a large number of elements that do not contain any sentiment data. These are elements such as links and user-tags.

This subsection describes several preprocessing techniques that can be used for Twitter data and selects the ones that will be used in the system.

5.1.1 General cleaning operations

First of all, there are several basic cleaning operations that can be used to normalize the tweets. These operations are not specific to Twitter data, as they are often used in a variety of data mining applications. As a start, all text can be converted to lowercase letters. Next, tabs, line breaks and new lines can all be converted to a single blank, followed by removing any forms of punctuation in the data. Finally, it is possible to remove any blanks present before or after the tweets and instances of two or more successive blanks can be converted to a single blank.

5.1.2 Twitter specific filtering operations

Tweets can contain a significant amount of unimportant and unusable data. First of all, URLs are elements often contained by a tweet. Naturally, URLs do not express sentiments and therefore have a negative influence on the quality of the data in this context, such that URLs can be removed from the data. Next, tweets often contain mentions ('@User', for example). Similar to URLs, these mentions do not contain sentiment information, meaning that they can be removed as well. According to Symeon Symeonidis et al. [13], removing URLs and user mentions has a positive effect on the accuracy of sentiment analysis. Giulio Angiani et al. [14] also showed that removing URLs and mentions, in combination with the general cleaning operations presented in 5.1.1, can increase the accuracy of sentiment analysis significantly. Therefore, both techniques will be included as preprocessing operations of the system.

Furthermore, there are several reserved words specific to Twitter that can be used to support tweets. ‘RT’ is an example of such a word. Reserved words have no sentimental value as well, which makes it possible to remove them, which was also proposed by Mengdi Li et al. [15].

Hashtags, however, often do contain sentimental values in tweets. The hashtag '#Happy’, for example, clearly contains a positive sentiment. Although it is proposed to remove hashtags completely by Giulio Angiani et al., it is believed that the sentimental value of these should not be discarded. Furthermore, the study performed by Symeon Symeonidis et al. did not remove any hashtags. Therefore, only the ‘#’ part of hashtags is removed.

5.1.3 Emoticons

Emoticons or emojis often contain much information about sentiments of social media posts [16]. Therefore, they should also be taken into account during sentiment analysis.

Giulio Angiani et al. suggest it is possible to define two categories for emoticons: a positive one and a negative one. If this is done, all instances of emoticons can be replaced by their textual counterpart. For example, the emoticon ‘:)’ would be replaced by the term smile_positive. This approach shows a clear increase in the accuracy of sentiment analysis.

5.1.4 Negations

Negations also have a significant influence on the sentiment expressed in a tweet. Where the tweet “The movie was not funny”, for example, represents a negative sentiment, the tweet “The movie was funny” represents a positive sentiment. Wrongly interpreting negations is one of the main causes of misclassification in sentiment analysis [14]. There are multiple techniques possible to handle negations in sentiment analysis.

One way to handle negations, is to leave in all negations without making any alterations to the text. The downside of this approach, however, is the fact that there exist many variations on the word 'not', such as 'won't', 'don't' and 'never'. These variations cause the model that predicts the sentiment of a tweet to need to handle multiple negation variations, some of which have a low occurrence frequency. This is not beneficial to the accuracy of the model.

Another way of handling negations comes in the form of replacing all different variations of occurrences of negations with the word 'not'. This technique is proposed by Giulio Angiani et al. and increased the accuracy of different prediction models by 0.15% up to 0.96%.

Furthermore, a third technique of handling negations was introduced by Perkins [17] and studied by Symeon Symeonidis et al. Following this technique, all instances of the word 'not' are found in a tweet, after which the next word is replaced with an antonym if there exists such an antonym. An example of this can be found in the tweet: “This book was not helpful”, which would be changed to “This book was useless”. It was shown that the technique slightly (0.2% to 0.4%) increases the accuracy of several machine learning models on certain specific data sets, but when used in the majority of the combinations of machine learning models and data sets, it did not have any effect or even decreased the accuracy of the model. The reason for this, according to Xia et al. [18], is that antonyms often represent a different sentiment.

Finally, Symeon Symeonidis et al. also introduced a fourth technique. This technique relies on finding all instances of the word 'no' in a tweet and adding a prefix NEG_ to all the words that come after it until the next punctuation mark. This approach tries to pay attention to the scope of negation in a sentence. An example can be found in the following tweet: “I dislike the cinema, because there is no popcorn.”. This would change into: “I dislike the cinema, because there is no NEG_popcorn.”. While the approach did show significant improvements in the accuracy of several sentiment analysis models (0.2% up to 1.3%), it also showed decreases in accuracy in others (0.1% up to 1.2%). This might be caused by the fact that the approach can significantly change the sentimental value of a statement. The statement “I like the cinema, because there are no barking dogs around”, expresses a positive sentiment. This statement, however, would be changed into: “I like the cinema, because there are no NEG_barking NEG_dogs NEG_around”. As around 27% of the words in this sentence are incorrectly marked as being negative, the predicted sentiment of the statement might change completely. The model also does not take variations of the word 'no' into account.

The technique of replacing variations of the word 'not' by 'not', shows the most promising balance between significantly increasing the accuracy of sentiment analysis models without risking a decrease in accuracy. Therefore, we decided to integrate this technique into the system.

5.1.5 Stop word removal

Stop words are words that generally occur in high frequencies in pieces of text. These words often do not contain any sentimental value and can, therefore, be removed from the tweets. Both the studies performed by Symeon

Symeonidis et al. and Giulio Angiani et al. showed an overall positive effect of stop word removal; when used in some prediction models, stop word removal slightly decreased the accuracy of the prediction model, but in the majority of cases this technique improved accuracy. Therefore, we chose to integrate the technique of stop word removal into the system.

5.1.6 Stemming

Stemming is a technique where certain variations of words are replaced by a basic variant. For example, the words 'greatly' and 'greatest' are replaced by 'great'. Giulio Angiani et al. found that stemming greatly increased the accuracy of sentiment analysis, while the results of Symeon Symeonidis et al. were mixed, with some increases and some decreases in accuracy.

Therefore, we chose to test and compare the accuracy on the system with stemming integrated and stemming not integrated to determine which strategy works the best for CITYSENT.

5.1.7 Spelling corrector

Misspelled words can change the sentimental value of words. Therefore, a spelling corrector can be used to correct these words. However, both Symeon Symeonidis et al. and Giulio Angiani did not achieve any accuracy improvements with spelling correction. It is important to note that the former expected that this was partially caused by the basic quality of the spellchecker that was used. The latter, however, did not report any similar expectations. Therefore, it was decided that a spelling correction mechanism should not be integrated into the system.

Elongated words are specific types of misspelled words. These are words where a character is wrongly, but often intentionally, repeated. These can be replaced by their normal counterpart. An example of this is "coooool", which would be replaced by "cool". Symeon Symeonidis et al. showed that this does not result in a significant improvement in the accuracy of sentiment analysis either. Therefore, this technique is also not integrated into the system.

5.1.8 Slang and abbreviations

Abbreviations of words is an element often present in messages on social media, as these abbreviations represent a certain form of informal language or slang. This is especially true for tweets, as Twitter restricts the number of characters that can be used in a single tweet. Abbreviations then provide Twitter users with a simple way of expressing a lot of information with a limited amount of characters.

Symeon Symeonidis et al. studied how replacing abbreviations can result in an accuracy improvement of sentiment analysis. This was also researched by Giulio Angiani et al. in combination with the spelling corrector introduced above. Both studies showed no significant increase in the accuracy. Therefore, it was decided to not include the replacement of abbreviations in the system.

5.1.9 Removing numbers

According to Symeon Symeonidis et al., numbers contained in text do not contain sentiment. Therefore, it was researched what the the effect of removing these from text would be. It was shown that this technique can have a positive effect on the accuracy of sentiment analysis, with improvements ranging from 0.1% to 0.3% on virtually all models this technique was tested on. Because of this relatively small but constant improvement, it was decided to also integrate number removal into CITYSENT.

5.2 Sentiment analysis

To determine whether Twitter users express a positive, negative or rather neutral feeling in their tweets, a technique commonly called sentiment analysis needs to be performed on these tweets. As the name suggests, this technique allows for the extraction of sentimental information from pieces of text.

The two most commonly used approaches for such analysis are a lexicon-based approach and a machine learning approach. The lexicon-based approach uses predefined lists of words annotated by sentiment scores, also called lexicons, to logically derive sentiment from a piece of text [19]. The machine learning approach, on the other hand, makes use of techniques from the field of machine learning to classify pieces of text into several sentiment categories [19], [20]. A hybrid approach, utilizing properties from both techniques, can also be used. [19]

In this subsection, these techniques are explained in more detail and their advantages and disadvantages are discussed, after which it is determined which will be integrated into the system.

5.2.1 Lexicon-based

In lexicon-based sentiment analysis, lists of words annotated by sentiment scores, also called polarity scores, are used to determine the sentiment of a piece of text. This can be done by summing the sentiment scores of the words present in the text such that the result indicates the overall sentiment score of the text [21]. The annotated words can be extended by the use of other elements such as annotated emoticons and several linguistic rules [19].

There are several ways of obtaining the required lexicons for lexicon-based sentiment analysis. The first of these is the manual creation of lexicons. Such creations require manually selecting words and annotating these with sentiment scores, which is time-consuming [21].

Another method comes in the form of a supervised approach. For this approach, a data set of sentences combined with labels indicating the sentiments they express is used. After preprocessing these sentences to remove words that cannot carry sentiment, a lexicon can be constructed from the words in the sentences according to the number of times they occur in sentences marked as positive and sentences marked as negative [21].

One last approach is creating a small lexicon manually and extending the lexicon using a set of rules. For example, a small lexicon of words can be expanded significantly if all antonyms and synonyms of these words are added to it.

The main benefit of lexicon-based analysis is that it is unsupervised, meaning that once a lexicon is obtained, no training data is required to train the system [19]. A downside, however, is that the language used in tweets is dynamic, as new expressions are constantly emerging [19], which makes it hard to have an up-to-date and accurate lexicon. Furthermore, the fact that polarity scores of text are calculated using the polarity scores of single words causes another problem, namely the impossibility of taking the context of words into account. This makes it hard to obtain a valid sentiment score from ironic statements or other statements that heavily rely on context.

Ogla Kolchyna et al. [21] studied the accuracy of a lexicon-based approach in comparison to a machine learning approach and a hybrid approach. The accuracy of this approach under a variety of different circumstances ranged from 51.33% up to 61.74%

5.2.2 Machine learning

The machine learning approach makes use of supervised learning algorithms to classify pieces of text into several discrete sentiment categories, as opposed to the continuous sentiment scores resulting from lexicon-based analysis. In order to make this possible, a data set is needed in which tweets are annotated by the sentiments they carry. There is a large number of supervised learning algorithms available, but the most popular ones for classification of pieces of text are Support Vector Machines [22], Naive Bayes [23] and Decision Trees [24] [21].

Decision Trees

Decision trees are classifiers constructed as a tree: in these trees, all non-leaf nodes represent decision points [24]. The branches that originate from these decision points represent the outcomes belonging to these points and the leaf nodes represent the class labels of the classifier.

The benefits of decision trees are that they are often relatively simple to interpret and that they can be used very well to perform classification operations on text. However, decision trees can also easily be overfitted on data [21].

Naive Bayes

Naive Bayes is a probabilistic classification model based on the Bayes theorem [23]. Using this theorem, several class labels, or hypotheses, can be defined. For every problem instance that needs to be classified, the probability of the instance belonging to a certain class label can be determined. The instance is then predicted to belong to the class with the highest probability.

The Naive Bayes approach is often used in sentiment analysis for its simplicity and effectiveness [21].

Support Vector Machines

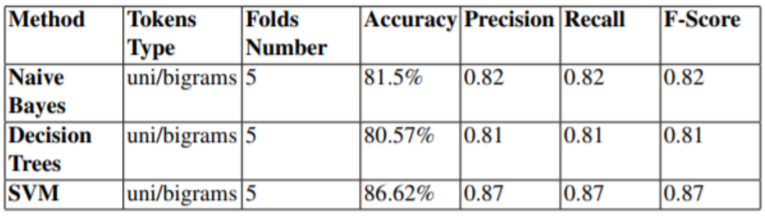
Support Vector Machines (SVMs) are used to find hyperplanes in N-dimensional spaces, where N is the number of features, such that the data points belonging to separate classes are separated by the hyperplane [22]. The goal of SVMs is to find the hyperplane, such that the margin (i.e. the distance between the two closest data points of the separate classes) is maximized in order to improve the confidence of predictions.

This approach has often been used in text classification scenarios where it was very successful, as SVMs are able to operate on large feature spaces. The downside of SVMs, however, is that they are not simple to understand and computationally expensive [21].

5.2.3 Model consideration

The different models and approaches have different advantages and disadvantages. A very important aspect of these models, however, is their accuracy when they are in sentiment analysis scenarios.

As was stated earlier, Ogla Kolchyna et al. [21] studied the accuracy of the different machine learning models in a sentiment analysis scenario. The results of this study can be found in table 8.

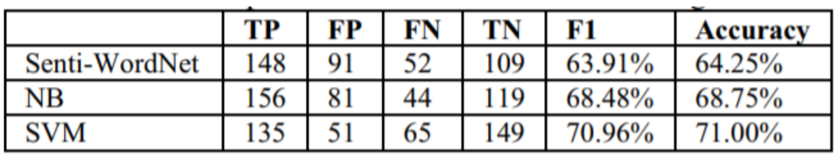


Method	Tokens Type	Folds Number	Accuracy	Precision	Recall	F-Score
Naive Bayes	uni/bigrams	5	81.5%	0.82	0.82	0.82
Decision Trees	uni/bigrams	5	80.57%	0.81	0.81	0.81
SVM	uni/bigrams	5	86.62%	0.87	0.87	0.87

Table 1: A comparison between the Naive Bayes, Decision Tree and SVM approaches[21]

From these results, it can clearly be seen that SVMs score the best on all used measures compared to Naive Bayes and Decision Trees. It can also be concluded that the SVM is significantly more accurate than the lexicon-based approach, because the accuracy of this approach ranged from 51.33% up to 61.74%, as stated earlier.

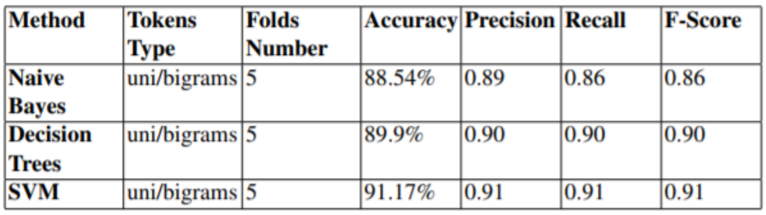
Similar findings were done by Hailong Zhang et al. [25], which are shown in table 2. Here, it can be seen that also in this study, the SVM approach was more accurate than both the Naive Bayes approach and the lexicon-based (Senti-WordNet) approach.



	TP	FP	FN	TN	F1	Accuracy
Senti-WordNet	148	91	52	109	63.91%	64.25%
NB	156	81	44	119	68.48%	68.75%
SVM	135	51	65	149	70.96%	71.00%

Table 2: A comparison between the Naive Bayes, Decision Tree and SVM approaches combined with the lexicon-based approach[21]

Ogla Kolchyna et al., however, also studied the accuracy of a hybrid approach, by using the sentiment score resulting from a lexicon-based analysis as an input feature to the supervised classification algorithms. The results of this can be seen in table 3.



Method	Tokens Type	Folds Number	Accuracy	Precision	Recall	F-Score
Naive Bayes	uni/bigrams	5	88.54%	0.89	0.86	0.86
Decision Trees	uni/bigrams	5	89.9%	0.90	0.90	0.90
SVM	uni/bigrams	5	91.17%	0.91	0.91	0.91

Table 3: A comparison between the lexicon-based, Naive Bayes and SVM approaches[25]

It can clearly be seen that the lexicon-based approach improves the accuracy of the supervised algorithms. In the same study, it was found that the lexicon-based score used as input feature also had the highest information gain score for the supervised algorithms, making it an important feature.

As stated earlier, SVMs are computationally expensive. However, as Twitter data is naturally streaming data, it is not likely that this will have a significant negative impact on the performance of the system.

For this reason and due to the high accuracy of SVMs compared to the alternatives, we decided to perform the textual sentiment analysis of CITYSENT using a SVM. To further improve the accuracy of the system, sentiment scores determined by a lexicon-based approach will be used as an extra input feature to the SVM.

5.2.4 Training data

To be able to perform classification tasks, the model presented in the previous subsection needs to be trained on performing sentiment analysis on pieces of tweets. For this task, training data is required. This data should consist of tweets annotated with a sentiment class label.

There are several options available to obtain this data. The first option would be to manually collect and annotate tweets with sentiment labels. This, however, is a tedious and time-consuming task. Therefore, another option would be to collect and annotate them in an automated way. According to Mendi Li et al. [15], this can be done by first collecting random tweets containing emoticons. Then, the tweets are divided into two categories: tweets containing a positive emoticon are marked as positive and tweets containing negative emoticons are marked as negative. Tweets from accounts of popular newspapers and magazines are used to form a set of neutral tweets. This approach, however, only works under the assumptions that the majority of tweets of newspapers do not express sentimental value and that positive and negative emoticons are only used in positive and negative context respectively. This last assumption prevents this approach from being used effectively to determine the sentiment of ironic or sarcastic tweets.

Furthermore, it is possible to use an existing data set of annotated tweets to train the classifier. The most popular of these is the sentiment140 data set [26]. This data set contains 600000 tweets annotated with a sentiment class. The annotations were obtained in a similar way as described by Mendi Li et al.: tweets containing ':)' were classified as positive and tweets containing ':(' were classified as negative. The main benefits of using this data set are the fact that it is publicly available, removing the need to collect a large number of tweets, and the fact that it is the data set used to train models in a large number of studies, often with positive results. The downside of this data set, on the other hand, is the fact that it only contains positive and negative tweets. Contrary to what is stated in the documentation of the set, it does not contain tweets marked as neutral. In practice, this might cause a bias, as also neutral tweets are classified as either positive or negative.

Another downside is the that the set does not contain emojis or emoticons, as these are all filtered out. According to Dane Hankamer and David Liedtka [27], this might be caused by the fact that many Twitter sentiment analysis studies were performed before the widespread usage of emoticons. Therefore, they introduce a method to introduce emoticons into sentiment analysis. They propose to use a mix of the Sentiment140 data set and a newly created emoji data set.

The emoji data set is created in several steps. First, random a set of tweets containing instances of the most popular emojis is collected by using the Twitter API. These tweets are classified by an existing sentiment analyzer and the resulting class is used as annotation. From the set that results from this, an equal amount of positive and negative tweets are selected and a new data set is created using these tweets.

Then, several classification models are trained on both the Sentiment140 set and the newly created set containing emoticons. This shows significant results in F1-scores, with an increase of almost 20 percent for Support Vector Machines.

For this reason, we decided to follow a similar approach as proposed by Hankamer and Liedtka to include emoticons in the analysis stage of CITYSENT. To also include neutral tweets, it was decided to follow the method proposed by Mengdi Li et al., using tweets of newspapers for this purpose. Here, it is also important to note that a lexicon-based score is used as an input feature to the classification model, as was explained earlier. As these lexicon-based scores range from -1 to +1 (or scaled versions of this range), with 0 being neutral, neutrality is also taken into account by the usage of this input feature.

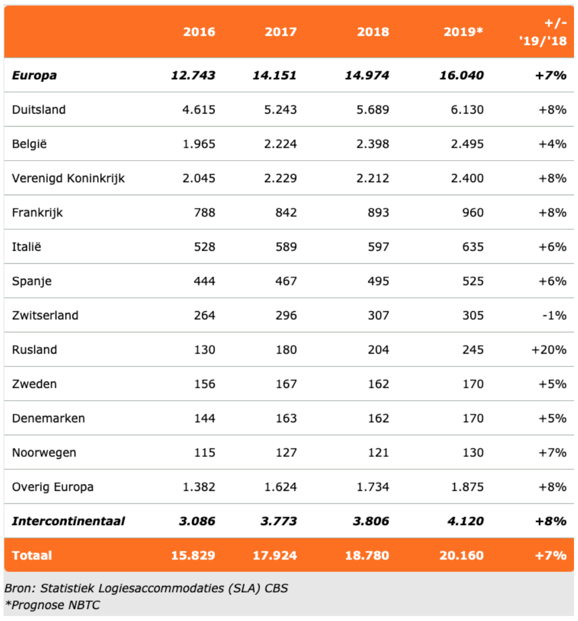
5.3 Language

Our main focus will be tweets with sentiment towards locations in the Netherlands and Flanders. So most of the tweets that we will mine will be written in Dutch, but we are also interested in the tweets of tourists and thus other languages. There are two challenges related to language: (1) "the majority of research papers utilize English as the only language of analysis" [28], but most of our collected data will be in Dutch and (2) when looking into the sentiment of tourists towards certain locations we should be analyzing multiple languages.

For the sentiment analysis of the Dutch tweets, we have two options: either translating the training data to Dutch or translating the tweets to English. Since we not only want to tackle the Dutch tweets, but also the tourists' tweets, the logical approach is to translate all tweets to English.

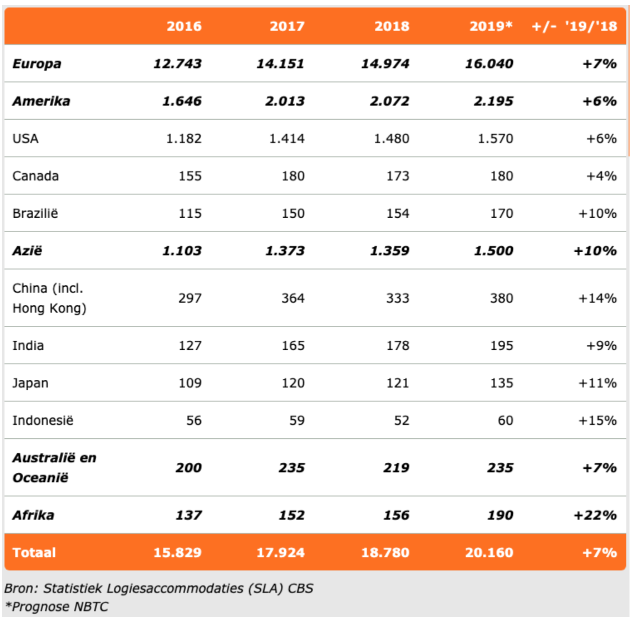
In table 4 a distribution can be seen of the countries of origin from the tourists visiting the Netherlands. Almost 22% of the tourists visiting the Netherlands come from countries where English is the native language (UK, USA, Canada and Australia).

Multiple studies use the Google Translate API for sentiment analysis on other languages than English [28], [31], [32]. In 2019, a study was conducted about the accuracy of Google Translate [30]. In table 5 a table is presented of multiple languages and their accuracy score on Google Translate. BLEU (Bilingual Evaluation Understudy) is an accuracy evaluation method for translations. "BLEU looks at the presence or absence of particular words, as well as the ordering and the degree of distortion—how much they actually are separated in the output" [33]. BLEU3 looks at the accuracy of translating foreign languages to English with Google Translate in 2019. The Tarzan score is the percentage of text that can be understood by humans. Tarzan2 also looks at the accuracy of translating foreign languages to English with Google Translate in 2019. The languages in table 5 together with English cover over 80% of the native languages of tourists visiting the Netherlands and



	2016	2017	2018	2019*	+/- '19/'18
Europa	***12.743***	***14.151***	***14.974***	***16.040***	***+7%***
Duitsland	4.615	5.243	5.689	6.130	+8%
België	1.965	2.224	2.398	2.495	+4%
Verenigd Koninkrijk	2.045	2.229	2.212	2.400	+8%
Frankrijk	788	842	893	960	+8%
Italië	528	589	597	635	+6%
Spanje	444	467	495	525	+6%
Zwitserland	264	296	307	305	-1%
Rusland	130	180	204	245	+20%
Zweden	156	167	162	170	+5%
Denemarken	144	163	162	170	+5%
Noorwegen	115	127	121	130	+7%
Overig Europa	1.382	1.624	1.734	1.875	+8%
Intercontinentaal	***3.086***	***3.773***	***3.806***	***4.120***	***+8%***
Totaal	**15.829**	**17.924**	**18.780**	**20.160**	**+7%**

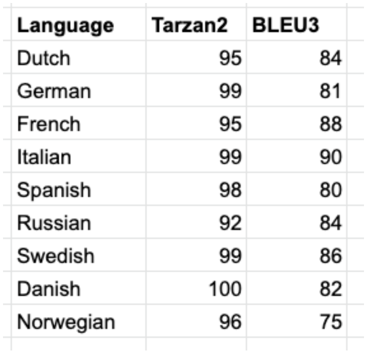
Bron: Statistiek Logiesaccommodaties (SLA) CBS
**Prognose NBTC*



	2016	2017	2018	2019*	+/- '19/'18
Europa	***12.743***	***14.151***	***14.974***	***16.040***	***+7%***
Amerika	***1.646***	***2.013***	***2.072***	***2.195***	***+6%***
USA	1.182	1.414	1.480	1.570	+6%
Canada	155	180	173	180	+4%
Brazilië	115	150	154	170	+10%
Azië	***1.103***	***1.373***	***1.359***	***1.500***	***+10%***
China (incl. Hong Kong)	297	364	333	380	+14%
India	127	165	178	195	+9%
Japan	109	120	121	135	+11%
Indonesië	56	59	52	60	+15%
Australië en Oceanië	***200***	***235***	***219***	***235***	***+7%***
Afrika	***137***	***152***	***156***	***190***	***+22%***
Totaal	**15.829**	**17.924**	**18.780**	**20.160**	**+7%**

Bron: Statistiek Logiesaccommodaties (SLA) CBS
**Prognose NBTC*

Table 4: Amount of tourists visiting the Netherlands in all accommodations (x1.000)[29]



Language	Tarzan2	BLEU3
Dutch	95	84
German	99	81
French	95	88
Italian	99	90
Spanish	98	80
Russian	92	84
Swedish	99	86
Danish	100	82
Norwegian	96	75

Table 5: Google Translate accuracy score [30]

as seen in the graph they all have high translation accuracy scores for Google Translate. Therefore, we decided to translate all the tweets into English using Google Translate.

6 Determining tweet characteristics

6.1 Location

Twitter gives users the possibility to geotag their tweets with the precise location they are posted from. About 1% of all tweets are geotagged, which might not seem like much, but it still gives us a good representation of the full tweet corpus. Moreover, it is a practical way to reduce the stream of data entering the system, making the processing of the data more sensible.

The alternative would be to infer locations from the tweets ourselves, in which case the system would extract the locations from the text of tweets. This is more commonly known as geoparsing or, more generally, named entity recognition. It would be a bit more involved than using geotagged tweets because interpreting what the text is actually about using natural language processing is a complicated process with some difficulties such as spelling errors and ambiguous names.

Moreover, there is one important distinction to be made between the two aforementioned approaches: searching tweets based on their geolocation gives us the sentiment *within* a place, while searching tweets based on their text gives us the sentiment *towards* a place. The latter will also yield fewer results, as people are often not expressing their sentiment towards a place in such an explicit manner.

Another major drawback of the text-based approach is its scalability. To recognize place names in the text, the system would need to make use of a gazetteer, which is a geographical index of place names with some corresponding information. To collect the tweets, the system would have to search Twitter for every possible place name in the gazetteer or compare every published tweet against the gazetteer. Either way, it would result in a tremendous overhead. By searching Twitter for every possible place name in the gazetteer, the system would be making at least one request for every place. A gazetteer that contains all place names is huge in practice, therefore this would not be feasible on such a large scale. The only way to solve this would be to use a predefined set of places that is not too large. However, using a smaller set of places limits the functionality of the application. For example, the user would not be able to search for any place they want, they would only be able to search for the predefined places. The other possibility of comparing every published tweet against the gazetteer has the issue that there are so many tweets that running a complicated entity recognition algorithm on each of them would not be realistic, again especially with a huge gazetteer.

In conclusion, the problems with the text-based approach could be solved by using a small gazetteer or filtering the incoming stream of tweets based on some other criterion beforehand. However, this always has a trade-off. Moreover, since natural language processing always comes with some uncertainty, the geolocation-based approach is also more reliable in practice. For the purpose of the application, it also seems more useful to derive an accurate, implicit sentiment rather than an explicit sentiment with far lower accuracy. Therefore, the geolocation-based approach is what will be used in the application.

6.2 Visual characteristics

Since Twitter does not share age nor gender, we will have to infer it ourselves. Luckily, most personal Twitter accounts have their face in their profile picture. To confirm this, a small script was run on 980 tweets from the Netherlands. For every tweet, it was determined whether there was a face in the profile picture by using a pre-trained face recognition model called face_recognition. As a result, 459 profiles (47%) were found to have faces in them. Looking at a random sample of 30 tweets, this result looks accurate. It is worth noting that face recognizers are not 100% accurate. According to GitHub, face_recognition has an accuracy of 99.38% for face verification. In order to do face verification, you need to detect a face and then find out if it is indeed the person you were referring to. Since we only use the face detection component of the library, it is safe to assume the accuracy will be similar or even higher. The other 53% of profiles were mostly news outlets or companies, which would not be useful for our database. News outlets and companies often have a (hidden) agenda when posting on Twitter and thus using their tweets will not give an accurate overview of the sentiment of certain places.

We will be using Convolutional Neural Networks, to infer age and gender. As of today, a paper written by Gil Levi and Tal Hassner [34] appears to be the de facto standard. Its accuracy is very high and it is also easy to implement.

6.2.1 Gender recognition

Gender recognition is easier to do than age recognition, since there are only two options: male or female. The hardest part is inferring the gender of young children, since a lot of gender-specific features are often not yet fully developed. The technique mentioned in the paper has an accuracy of 77.8 +- 1.3. But when image cropping is used, it is bumped up to 85.6 +- 1.4. Since the script we used for detecting faces already gives us the correct coordinates of a face, we will also use image cropping for gender and age recognition.

6.2.2 Age recognition

Exact age recognition is very hard to do, since a lot of things can influence how old people look (make-up, lighting, etc.). Thus, instead of inferring the exact age, we will use classification. Using classification (discrete) instead of regression (continuous) will make the problem of age recognition easier to solve, while still getting useful results. Gil Levi and Tal Hassner use the age brackets {0-2}, {4-6}, {8-13}, {15-20}, {25-32}, {38-42}, {48-53}, {60+}.Note that some ages are missing, this is done to make the estimation of age easier. When used with cropping, the accuracy was 45.1 +- 2.6 and 79.5 +- 1.4 of the estimations were correct or one age bracket off. So in general, estimations for specific age brackets are not very accurate, but when one bracket off errors are allowed, accuracy is acceptable. For our project, such precise age brackets might be a bit too precise and since the age is inferred anyways, we have decided to go for a bit more generic brackets. Ideally, we would classify people into the following brackets: child {0-13}, teen {15-20}, young adult {25-32}, adult {38-53}, senior {60+}. We still use the same thresholds, but are combining some brackets together. Combining brackets allows us to have higher accuracy, since a high percentage of estimations is one bracket off. For example, if the recognizer returns the bracket {4-6}, but the child in the picture belongs to {8-13}, the answer is wrong. In our model, the answer is still correct because the child will be classified as a 'child' {0-13}. According to the age estimation confusion matrix made by Levi and Hassner, the hardest ages to estimate are {15-20}, {38-42} and {48-53}. [1]

[1]The pictures used in the paper are sometimes blurry or of low resolution, which influences accuracy a lot. Twitter profile pictures are most of the time of acceptable resolution and not blurry, so the values of the confusion matrices will in our case most likely be higher.

	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60-
0-2	**0.699**	0.147	0.028	0.006	0.005	0.008	0.007	0.009
4-6	0.256	**0.573**	0.166	0.023	0.010	0.011	0.010	0.005
8-13	0.027	0.223	**0.552**	0.150	0.091	0.068	0.055	0.061
15-20	0.003	0.019	0.081	**0.239**	0.106	0.055	0.049	0.028
25-32	0.006	0.029	0.138	0.510	**0.613**	0.461	0.260	0.108
38-43	0.004	0.007	0.023	0.058	0.149	**0.293**	0.339	0.268
48-53	0.002	0.001	0.004	0.007	0.017	0.055	**0.146**	0.165
60-	0.001	0.001	0.008	0.007	0.009	0.050	0.134	**0.357**

Table 6: Confusion matrix made by Levi and Hassner (top is actual age, left is predicted age)[21]

	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60+
0-13	**0.982**	**0.943**	**0.746**	0.179	0.106	0.087	0.072	0.075
15-20	0.003	0.019	0.081	**0.239**	0.106	0.055	0.049	0.028
25-32	0.006	0.029	0.138	0.510	**0.613**	0.461	0.260	0.108
38-53	0.006	0.008	0.027	0.065	0.166	**0.348**	**0.485**	0.433
60+	0.001	0.001	0.008	0.007	0.009	0.050	0.134	**0.357**

Table 7: Confusion matrix made from combining certain age brackets[21]

If you compare the new matrix to the old one, confusion will (in theory) be lower for classifying young children and old adults. However, confusion is still high when predicting {15-20}. After a discussion with the client, we decided to instead of age brackets, we divide people into three categories: child {0-13}, adult {15-53} and senior {60+}. This led to the following confusion matrix.

	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60+
Child	**0,982**	**0,943**	**0,746**	0,179	0,106	0,087	0,072	0,075
Adult	0,015	0,056	0,246	**0,814**	**0,885**	**0,864**	**0,794**	0,569
Senior	0,001	0,001	0,008	0,007	0,009	0,050	0,134	**0,357**

Table 8: Confusion matrix made with 3 categories: child {0-13}, adult {15-53} and senior {60+}[21]

Using three categories has its trade-offs, you increase correctness at the expense of accuracy. For example, a person of age 15 will probably behave a lot differently than an adult of 32, but they both get classified as 'adult'. Still, after a discussion with the client, it was decided that higher correctness was more important. Otherwise, the research the client wants to do would be based on false data.

The accuracy of predicting 60+ is still very low, but it is worth noting that not a lot of seniors use Twitter (only 7% of {65+}, in contrast to 38% of {18-29})[2] [35]. Since the number of tweets posted by seniors will be relatively low compared to adults, having half of senior tweets be classified as tweets made by adults will not influence the adult sentiment as much. The senior tweets that do get classified correctly can then still give useful insight into the senior sentiment. We also think that the confusion is mostly for people around 60, not 70+, but this can only be confirmed in the implementation phase. If confusion is indeed mostly for people around 60, the misclassification is not a big deal, since people around 60 could also be treated as adults.

Another way to lower confusion is to look at the way people type and infer age from that data. This can be done with good accuracy, especially for young people, since they often have a different kind of 'slang' than older people. Using inference through text in combination with visual recognition would maybe allow us to use age brackets and thus increase accuracy. Inferring age through language is only accurate for young people since older people tend to type the same way, thus only using language-based inference instead of visual-based inference is not a good option. Models have already been made that can make such estimations, but since this might take a while to implement, it is only useful to look further into age inference through text when other, more necessary, features are implemented. [36]

[2]These figures are from the United States in 2019, so the percentages in the Netherlands might be slightly different.

6.2.3 Activity recognition

Another thing that can be deduced from pictures, is which activity was done during certain sentiments. For example, it might be useful to know that people are often happier when drinking with friends in certain places. Or maybe a certain park shows positive sentiment mostly during sport. While this data is very interesting and valuable to have. Deducing activities from pictures is hard and often not very accurate, even for state of the art recognizers such as Google Vision. So creating an activity recognizer ourselves is probably out of scope for this project. The use of services, like Google vision, that provide activity recognition could be a valuable addition to our tool, but just like age inference through text, this is something to look into once features with higher priority are done.

7 Data correlations

A possible extension to our tool would be to search for correlations. "Correlation is any of a broad class of statistical relationships involving dependence." [37] For example, if the user is interested in seeing whether the number of parks in a city affects its sentiment, the user can upload their own data set containing the number of parks per city. The system will then analyze the cities' sentiment scores and calculate how likely it is that the number of parks influences the sentiment. Besides the clients being able to upload their own data sets to search for correlations, we would also provide them with some example data-sets to play with.

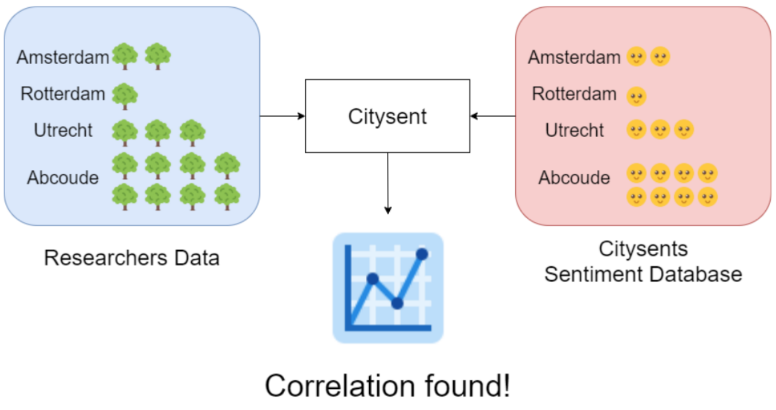


Figure 3: Correlation example

You could argue that instead of adding a data set with information about parks you could also search for tweets that mention parks. Brown et al. [38] used this method to find a correlation between sentiments in tweets and the stock market. Tweets were filtered for certain stock symbols, so we could filter on factors like parks, criminality and other things we think could influence people's sentiment. But for the scope of our project, filtering on topics is not what we want. We look into the sentiments of people in certain locations, assuming on the scale we are looking, sentiments are influenced by the locations. Rather than looking into sentiments of people towards certain locations. By looking at sentiments of people in a certain location, you also include the sentiments of people in a city that do not express their feelings towards a city and subconscious sentiments. For example, a tweet that is very positive about make-up posted in Rotterdam, will increase the score of Rotterdam. Thus filtering tweets on certain subjects to extract correlations is not what we want.

What we want is to find the correlation between the sentiment data and some input data set. Examples of interesting correlations for CITYSENT are:

- Rural vs urban areas
- Climate/weather
- Crime
- Negative events (e.g. COVID-19)

If you have more than two features is it very hard to calculate the correlation [39]. So in our tool we will only be calculating the correlation between the sentiment and one input feature. Which means the client will only be able to add data with two columns: one being the location and one being the feature they want to find a correlation on. The locations are matched with the correlations in our database (over a certain period of time entered by the client), then the correlation between the sentiments and the imported feature is calculated.

There are different types of correlations: linear (also monotonic), monotonic (non-linear) and non-monotonic. For instance, crime would probably be a linear or monotonic correlation with the sentiment. As crime increases in a certain area, so does the negative sentiment. But if you look at the weather, people do not like it if it is very cold, nor if it is very hot. So that would be a non-monotonic correlation. So our tool should be able to handle both linear, monotonic and non-monotonic correlations.

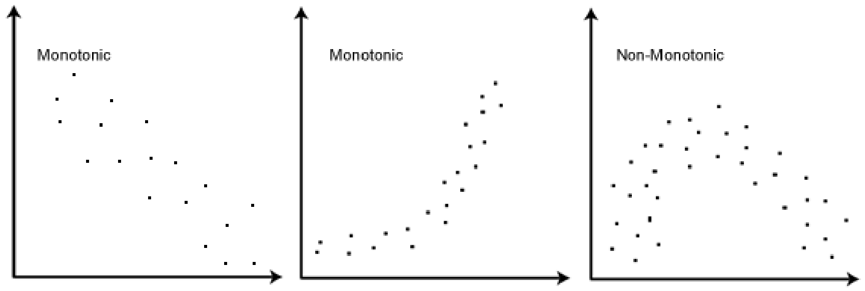


Figure 4: left to right: linear (also monotonic), monotonic (non-linear) and non-monotonic[40]

There are multiple correlation coefficients, Pearson's Correlation is most commonly used [41]. But with Pearson's Correlation, only linear correlation can be calculated. When searching for the correlation with ordinal data, Spearman's rank-order correlation or a Kendall's Tau Correlation are recommended [42]. Croux et al. [43] studied the robustness by means of the influence functions and gross-error sensitivities of the Spearman and Kendall correlations. They found that the Kendall correlation is more robust and even a little more efficient than Spearman's correlation. But both of these correlations are only feasible with monotonic data, since Kendall's correlation "measures the strength and direction of association that exists between two variables [44]. We found that there are no correlation coefficients for non-monotonic data. When calculating Kendall's correlation on non-monotonic data, the different directions will neutralize the coefficient.

Since it is not feasible to calculate the correlations for non-monotonic data, we would display the plotted data. By plotting the data, the client can draw conclusions about the trustworthiness of the returned correlation coefficient.

It is also important to keep in mind that correlation does not always imply causality. As explained in an example by [39]: "People aboard the Titanic did not die because they were male and aged 28. Rather, a large number of them died because officers were saving “Women and Children First”."

Concluding, we would be using Kendall's Correlation Coefficient to calculate the correlations between the sentiments and the input data. We will display a plotted graph of this data to give the client some insight into the data and to check if there is a monotonic relation.

8 Back-end

The back-end is the part of the system where everything except the user interaction will be happening. It will not only process all incoming tweets but it also needs to persist data and serve results to the user. The user will not be directly interacting with the back-end, but they will use an interface that queries the back-end for data. This way the server can do all the heavy lifting while the client only needs to worry about displaying the data to the user. The server needs to always be active as it needs to process incoming tweets at all times. Of course the client should always be able to reach the server as well.

In the very early phases of development, our own computers can run the server to fetch some tweets and verify some functionality of the application. However, soon it will be necessary to have a more complete, persisted set of data, built up over some more time. For this reason, it is important to get an external server fetching and persisting tweets up and running as early in development as possible. Our computers can only do this while the program is running and the fetched data is not centralized if everyone has their own set of tweets stored on their computer.

There are essentially four types of hosting we can use for the back-end: shared hosting, a virtual private server (VPS), a dedicated server, or cloud hosting [45]. With shared hosting, one server is shared with lots of users. Everyone gets the same operating system and installed software, so it does not provide much flexibility or control. On the other end, there are dedicated servers. In this case, you get the entire server for yourself, giving you all the control you need. However, a dedicated server is very expensive. To bridge this gap, there are VPSs. With a VPS, a physical server is split into multiple virtual servers that can each run their own software. Lastly, there is cloud hosting, which is similar to a VPS. The key difference is that it allows you to use different physical servers rather than just one, which is the case with a VPS. As cloud hosting allows a bunch of users to use a bunch of servers, it is highly scalable. Moreover, it allows you to use exactly what you need and only pay for what you use, making it a lot cheaper than a dedicated server.

For this application shared hosting would not suffice, as we need to have the back-end running at all times to process new tweets. Therefore, a VPS or cloud hosting is the best solution. The biggest cloud providers are Amazon Web Services, Microsoft Azure, and Google Cloud Platform [46]. However, there are also smaller VPS and cloud providers available that can be cheaper. It seems that the TU Delft also provides us with a "project server" VPS that hopefully suits the needs of this application.

As Twitter naturally gives us a stream of data, it is quite a logical idea to apply stream processing to get results. Stream processing lets us easily process every newly incoming tweet pushed to the system by the Twitter Streaming API. One platform that allows us to do this is Apache Kafka, a distributed stream processing platform that lets us process the data, persist the results, and query them, all in real-time [47]. In Kafka, all data is ordered into topics, which are logs of events. These events are pushed into the system by a "producer" (e.g. the Twitter Streaming API). In turn, a "consumer" can subscribe to these event streams in order to get notified about new events and process them. Because Kafka stores all these logs on disk, there is no need for a separate database [48], as the logs form the source of truth and Kafka provides ACID guarantees [49].

The alternative to stream processing would be batch processing the tweets. In this case, the back-end would run a job each day or hour, for example, retrieve all the tweets published in that time segment, and process them, after which the results would be stored in a traditional database. In contrast, stream processing gives us the advantage that all the data in the application is real-time and there is no system downtime for processing an enormous batch at once. Therefore, we aim to use Kafka to process and persist the data. However, because stream processing seems to be a slightly more involved process and the ability to use Kafka depends on the environment that we use, we will leave the use of Kafka subject to change.

9 User interface

The user interface is the second main component of the system and is responsible for interacting with the user and presenting the analyzed data in a structured format. Because the goal of the tool is for the user to be able to perform visual analysis of sentiment data, the user interface is important and needs to be both interactive and user friendly. Making a user interface for visual analytics tool is a challenging task by itself and *visual analytics* as a whole has its very own scientific research field. According to Daniel Keim et al. [50] visual analytics "combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex data sets".

Visual analytics differs from *information visualization*, which only concerns itself with displaying data to the user, by allowing the user to directly affect the data analyzing process. Interactivity is an important aspect of our tool to prevent *information overload* and facilitate the user's *knowledge discovery*. Information overload is a concern because we will be dealing with a large, complex data set with many variables and statistics. Preventing information overload will allow the user to concentrate better on the task at hand and perform better visual analysis. Preventing information overload will be done by allowing the user to choose which parameters they want to do analysis and comparison on in the form of filter options. Lastly, knowledge discovery is facilitated by allowing the user to play with the data and tweak parameters directly used for further analysis. As a result, the final analysis and result provided by our tool is tailored specifically to the user's interest and will provide the user with only useful, relevant data to base their decision on.

The specific details of the user interface will need to be established through user testing in development, but based on visual analytics research we can establish a rough outline of the functionalities of the interface as well as how certain types of analysis is best displayed visually to the user.

We will base the user interface functionalities on Shneiderman's visual information seeking mantra "Overview first, zoom and filter, then details-on-demand" [51], as is the de facto standard for many analytics-driven visualization systems [50]. What this means for the tool specifically, is that the tool will first provide the user with a general *overview* of data (e.g. ranking cities by their sentiment score and geographical heat maps).

The second step, *zoom and filter*, does not necessarily need to be taken literally. In this step, the user will be able to specify the data they are interested in for further analysis (e.g. zooming in on a geographical heat map and filtering on time, location, language, age, etc. to discard irrelevant information).

The third and last step is self-explanatory: take for example the sentiment scores of cities, the user could be able to click on a specific city to see the breakdown of its sentiment score (e.g. sentiment per neighborhood or which words/emojis contributed the most to positive/negative sentiment). The sentiment breakdown could be static (e.g. a pie chart) or dynamic to show temporal changes (e.g. a stacked area chart).

In addition to the three steps in the mantra, Shneiderman [51] added 'Relate', 'History' and 'Extract' as natural expansions. Therefore, if time allows, we will also incorporate sentiment comparison ('Relate'), cached filter options ('History') and data export in the form of reports ('Extract') as possible options in the user interface.

10 Other aspects

10.1 Programming language & frameworks

The language that will be used the most, is Python. Python has a lot of libraries and APIs for machine learning, such as PyTorch and OpenCV. Kafka, the streaming platform that we will be using, also supports Python. Furthermore, the NumPy library will be used for calculations done on the data (for example, calculating correlation coefficients). For the application back-end we will also use Flask[3] and for the front-end we will use React[4]. Flask was chosen because most of our code will be in Python. Flask is also in Python and will thus be easily compatible. Moreover, it is more lightweight than the alternatives, such as Django. React is chosen because it is easy to use and has extensive documentation. Since React is back-end agnostic, it does not matter that the back-end is in Python but the front-end in JavaScript.

10.2 Testing

To automate the testing, we will use the Python unit test framework. These unit tests must reach a coverage of 80%. Moreover, there should also be some system/integration tests to test the overall functionality of the system. To test the front-end of the application we will make use of Mocha, a JavaScript testing framework.

Besides testing the functionality of the code, we will also use static analysis tools to make sure the code base is maintainable and less prone to errors. We aim to use these tools to cover every part of the codebase:

- Flake8 (Python linter)
- ESLint (JavaScript linter)
- Stylelint (CSS linter)
- html-linter (HTML linter)

The static analysis tools will also make sure that all code conforms to certain style conventions. All Python code will conform to the PEP8 standard and all JavaScript code will conform to the AirBnB standard. In principle, all errors raised by the static analysis tools must be solved. However, in case there is actually a good reason to ignore an error, we will allow this.

The complete automated testing suite should pass at all times. Moreover, it should run on a continuous integration server to verify that the code in our Git repository is always functioning properly.

Besides automated testing, we also plan to perform a user test to verify that the user interface is intuitive to use. After conducting a user test, we can improve any flaws found here.

To test the machine learning models we will use training, validation, and test sets (this is standard practice) to check whether the model has been trained correctly and did not overfit on the data. We want to enforce an accuracy of at least 65% on the test set but aim for higher accuracy. This threshold is based on numbers found in similar research.

11 Conclusion

While there already exist tools that extract sentiment from social media, their focus lies mostly on city planning and not on general research. Tools that do focus on general research, such as Social Glass, moved their use cases from sentiment analysis to things like crowd management. Because of this, CITYSENT can add value by analyzing sentiment data on social media and making this data available to researchers who can use it for any research they require.

For obtaining the data, we decided to use the Twitter API, as Twitter has the least restrictive API of all the large social media platforms and has fairly evenly distributed demographics in terms of users. The Twitter API not only makes data collection easier, but it also provides valuable information that we otherwise could not get

[3]This might be subject to change, when we progress into the development phase.

[4]For now, we will settle with React. But if during the programming phase we have certain needs that React cannot satisfy, we might use other tools alongside it.

from scraping. To make sure that we handle the data we retrieve from Twitter responsibly, we will make sure to anonymize the stored data in compliance with both Twitter's ToS and the GDPR.

After the data is obtained, the system will perform a process called sentiment analysis on the tweets. Before this, however, the tweets need to be preprocessed to improve their quality. We have chosen a series of preprocessing techniques to be applied. These techniques include basic textual cleaning, the removal of URLs, user mentions, reserved words and the '#' sign from hashtags. Emoticons will also be replaced with a textual term representing them. Furthermore, variations of the word 'not' will be replaced by 'not' and numbers are removed from tweets. Misspelled and elongated words, on the other hand, will not be removed and the same holds for slang and abbreviations, as these techniques did not show signs of significant accuracy increases. Lastly, the technique of stemming will be applied to research its effect on the accuracy of CITYSENT.

For the sentiment analysis itself, different options were compared to each other. These options include methods based on lexicons, Naive Bayes models, Decision Tree models and Support Vector Machine models. From this, we concluded that an approach utilizing both Support Vector Machines and lexicons will be the most effective. In terms of languages, we will translate all tweets to English using Google Translate, since we will use English training data.

Determining the location of a tweet will be done using the provided geolocation of the tweet. This also reduces the stream of tweets entering the system. For age and gender detection we will be using Convolutional Neural Networks, created by Gil Levi and Tal Hassner. To increase accuracy for age estimations, there will only be three classes: child, adult and senior.

For the computation of correlation between the sentiment and the input data set, we would be using Kendall's Correlation Coefficient. We would also display a plotted graph of the data, such that the client is able to look into how the correlation coefficient was formed.

The back-end will be running on a virtual private server or on cloud hosting, as shared hosting is too limited for the application. It will use Kafka to process the incoming stream of tweets and persist all the data. The frond-end will be based on the functionalities previously identified to be important to any user-friendly visual analytics application. These functionalities include *overview*, *zoom and filter* and *details-on-demand* with the possible extensions of *extract*, *history* and *export*.

A number of libraries and frameworks will be used and most of the code will be written in Python. The front-end will be written using React.

Combining all these things will result in an easy to use, yet effective tool for sentiment analysis. Researchers can quickly find possible correlations between certain elements and sentiment, without doing a lot of data processing themselves. CITYSENT aims to contribute to an accurate yet extensive overview of sentiments and aid researchers in their quest for creating better cities.

A Project Planning

The project planning will give an overview over the project assignment, setup and planning.

A.1 Project assignment

This section will describe the assignment of the project. The first subsection describes what led our client to the project. The second subsection defines the project assignment itself. Finally, the product that we will create and its requirements are explained in the third subsection.

A.1.1 Project environment and goal

This project is commissioned by researchers of the faculty of Architecture and the Built Environment of the TU Delft. More specifically, the department of Urbanism, which performs research related to urban design, spatial planning, landscape architecture and environmental modelling [52].

The goal of this project is to create a tool that can be used to perform visual analytics on the sentiments people have towards certain locations and cities, as this information can provide researchers with insights that can help improve the quality of these urban environments.

A.1.2 Assignment specification

Our final product should fulfill the aforementioned goal. Therefore, a system needs to be created that fetches data from twitter including the geolocations of tweets. This data should be processed and analysed in order to obtain information about the sentiments people have while being at certain locations. We will also be analyzing tweets for the age and gender of the author, based on their profile picture. This data needs to be stored in a database, such that it is possible for the client to use it for personal research, through an interface we provide. In this interface the client will be able to see the different sentiments per age group, gender and location (e.g. rural vs urban areas).

A.1.3 Final project requirements

The final product will be a system that meets the described requirements and demands as specified below. We have used the MoSCoW method for prioritization of the requirements. The method is often used to help key stakeholders understand the significance of initiatives in a specific release [53]. The MoSCoW method divides the project in four requirements: must haves, should haves, could haves and won't haves.

Must haves

- The system must gather tweets from Twitter based on geolocation
- The system must support Dutch language posts to be processed
- The system must be able to calculate sentiment scores based on textual Twitter data
- The user must be able to search for a place contained in the database
- The system must be able to geocode[5] places (e.g. cities, neighbourhoods, etc.) from queries and show the corresponding sentiment
- The system must be able to store the sentiment and location data of tweets in a database

Should haves

- The system should do sentiment analysis based on hashtags
- The system should support English language posts to be processed
- The system should be able to compute the sentiment scores of cities and their neighbourhoods over time
- The system should, with reasonable accuracy, estimate the age of the Twitter author based on the profile picture
- The system should, with reasonable accuracy, estimate the gender of the Twitter author based on the profile picture

[5]Geocoding is the process of converting a string of text to its corresponding coordinates.

- The user should be able to make comparisons of sentiment between different categories of people (e.g. young vs old, male vs female)
- The user should be able to make comparisons of sentiment between different categories of places (e.g. urban vs rural areas)
- The system should have a map displaying an overview of the sentiment in different areas

Could haves

- The system could do sentiment analysis based on emoji/emoticons
- The system could extend data mining to other languages next to Dutch and English
- The system could be able to support data mining on multiple social media platforms
- The system could scan images to see which activities get posted about
- The system could be able to search Twitter for place names and variations thereof in tweet texts
- The system could predict the age of the user based on the tweet text
- The system could be able to infer the topics of a tweet
- The system should be able to visually display the sentiment scores of cities and their neighbourhoods over time to the user
- The system should be able to display the breakdown of a sentiment score to the user (e.g. the words/emojis/topics that contributed the most to a negative or positive score)
- The system could be able to combine sentiment scores with static data about the location (e.g. statistical data about a city like demographics, crime rate) to calculate whether there is a likely correlation between the sentiment score and the data
- The user could be able to upload their own static data and the system will calculate whether there is a likely correlation with the sentiment score

Won't haves

- The system won't support data mining of Facebook and Instagram (difficult to obtain legal access)

A.2 Project setup

There will not be physical meetings, in order to comply with the rules imposed on us regarding the COVID-19 crisis. It is thus very important to have a good communication framework, so that everyone can work from home. Methods and tools will need to be different from what we are used to, so that a good 'working from home environment' can be facilitated.

A.2.1 Methods

Scrum is used in order to keep pace and reach the project goals in time [54]. There will be weekly sprints in which we evaluate the work that has been done in the previous sprints and set goals for the current sprint accordingly. At the end of each sprint, there is a meeting with the client to evaluate the current state of the product. Once a week there will be a meeting with the coach, in which we can ask relevant questions and give updates on our progression. Every Monday morning we start the week with an organised meeting with an agenda and one of us taking notes. There will be a video call with each other at the beginning and the end of the day. More video calls can be done if necessary.

A.2.2 Tools

Since there will be no physical meetings due to COVID-19, a number of different tools will be used for communication:

- Jitsi will be used for group meetings with video calling
- Zoom will be used for meetings with the client
- We will all be available from 9.00-17.00 on Discord, to ensure quick communication and responses between project members
- Mattermost will be used for text-based communication with the group coach
- WhatsApp will be used for text-based communication with the client and within the group itself

A.2.3 Techniques

To achieve the project goals, multiple techniques will be used.

- Either connecting to an API or using a data scraper to extracts text and pictures from social media
- A combination of Natural Language Processing and lexicon-based text analysis to extract sentiment data from text data
- Machine learning on pictures to estimate age and gender

To ensure code quality, we will make use of testing and static analysis. Most of the testing will be automated by means of unit tests and where possible some integration/system tests. Ideally, we will also perform some user tests to confirm the usability of the end product. To ensure that the code conforms to certain standards and is less prone to errors, static analysis tools will be used.

A.2.4 Planning

The project is split up into three phases: research, development and delivery. The research phase stretches over the first two weeks of the project. This phase involves getting to know the coach and client, creating the project plan and writing the research report. The development phase will last six weeks, containing six sprints of one week each. The final two weeks will be the delivery phase, where the final report and presentation will be made.

The table below gives an overview of the planning for the project.[6]

Date		Description
Week 1	22-04	Deadline project planning
Week 2	29-04	Deadline draft research report
	01-05	Deadline final research report
Week 3		Project setup finished
		Setup for obtaining tweets finished
Week 4		Preprocessing of Tweets finished
		Age and gender prediction finished
Week 5		Sentiment classification finished
		Tweet streaming/storage finished
		Query to geocode conversion finished
Week 6	29-05	Deadline SIG 1
Week 7		UI finished
		User tests
Week 8	12-06	Deadline SIG 2
Week 9	16-06	Deadline final report
Week 10	23-06	Presentation

A.2.5 Contract

As discussed in the first meeting with the client, they will draw up a contract as soon as possible. Because the client would first like to conduct some research themselves using the application rather than making it public immediately, we will probably sign an NDA for a relatively short amount of time. The offered stipend will be included in this contract as well.

A.3 Quality assurance

This section will describe five aspects of the project that need to be taken into account to ensure the quality of the end result: functionality, usability, efficiency, maintainability, and security/privacy. The following subsections will briefly explain the essence of these aspects and how they should be accounted for.

A.3.1 Functionality

With the goal that the client has in mind, it is most important that the product functions as desired. This means it should comply with all the requirements described earlier.

[6]Planning is subject to change, since we'll have to discuss a few specifications with the client

A.3.2 Usability

Because the application is ultimately not going to be used by computer scientists but directly by a user, the application should have a highly usable user interface. It is important that this is taken into account when building this interface, such that the end product will be easy to use for its intended target group. This could be achieved by conducting user tests and using established heuristics.

A.3.3 Efficiency

The application will be dealing with a vast amount of data, therefore it is important that this is done in an efficient manner. Moreover, this will improve the usability of the product, as it ensures that the user can directly interact with the system. Having an efficient solution will also result in an end product that is cheaper once enrolled, as it will require less computation.

A.3.4 Maintainability

To allow the product to be extended and used in the future, the codebase needs to be maintainable. Therefore, the code should be properly documented, tested, and conform to certain style conventions. It is also important that the project has a modular structure to ensure its scalability. Given these prerequisites, other developers will be able to continue working on the project.

A.3.5 Security and privacy

The application will retrieve a large amount of data from public sources, such as social media platforms. However, to be able to guarantee the privacy of the users of these platforms, only aggregated user data will be stored persistently. This means that the fetched data will be processed immediately, such that only the result needs to be stored.

The largest security risk of the application results from the ability to interact with the system via an interface. Although all of the data in the database is open for the client to use, the application might eventually also be used by other parties that might not have these same access rights. Therefore, all forms will be sanitized and evaluated for the containment of malicious statements.

References

[1] J. Isaak and M. J. Hanna, "User data privacy: Facebook, cambridge analytica, and privacy protection", *Computer*, vol. 51, no. 8, pp. 56–59, 2018.

[2] J. Perriam, A. Birkbak, and A. Freeman, "Digital methods in a post-api environment", *International Journal of Social Research Methodology*, vol. 23, no. 3, pp. 277–290, 2020.

[3] M. Zimmer, ""but the data is already public": On the ethics of research in facebook", *Ethics and information technology*, vol. 12, no. 4, pp. 313–325, 2010.

[4] E. Kernel, *Three biggest legal cases about data scraping*, Dec. 2019. [Online]. Available: https://jaxenter.com/data-scraping-cases-165385.html, (Accessed on 30/04/2020).

[5] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, V. Sanchez, and H. Perez-Meana, "A web scraping methodology for bypassing twitter api restrictions", *arXiv preprint arXiv:1803.09875*, 2018.

[6] Newcom Research, *Social media onderzoek 2020 - Newcom Research*, https://www.newcom.nl/index.php?page=socialmedia2020, (Accessed on 28/04/2020).

[7] Facebook, *New Facebook Platform Product Changes and Policy Updates*, https://developers.facebook.com/blog/post/2018/04/24/new-facebook-platform-product-changes-policy-updates/, (Accessed on 30/04/2020),

[8] Instagram, *Platform Changelog • Instagram Developer Documentation*, https://www.instagram.com/developer/changelog/, (Accessed on 30/04/2020), 2019.

[9] Pew Research Center, *Social media usage in the U.S. in 2019 | Pew Research Center*, https://www.pewresearch.org/fact-tank/2019/04/10/share-of-u-s-adults-using-social-media-including-facebook-is-mostly-unchanged-since-2018/, (Accessed on 28/04/2020).

[10] P. M. Schwartz and D. J. Solove, "Defining 'personal data'in the european union and us", *Privacy & Security Law Report. http://docs. law. gwu. edu/facweb/dsolove/files/BNA-Schwartz-Solove-PII-US-EU-FINAL. pdf*, 2014.

[11] The European Union, *What is considered personal data under the eu gdpr?* [Online]. Available: https://gdpr.eu/eu-gdpr-personal-data/, (Accessed on 30/04/2020).

[12] S. Zhang, "Information enhancement for data mining", *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, vol. 1, pp. 284–295, Jul. 2011. DOI: 10.1002/widm.21.

[13] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis", *Expert Systems with Applications*, vol. 110, pp. 298–310, 2018, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.06.022. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418303683.

[14] G. Angiani, L. Ferrari, T. Fontanini, P. Fornacciari, E. Iotti, F. Magliani, and S. Manicardi, "A comparison between preprocessing techniques for sentiment analysis in twitter", in *KDWeb*, 2016.

[15] M. Li, E. Ch'ng, A. Chong, and S. See, "The new eye of smart city novel citizen sentiment analysis in twitter", Jul. 2016. DOI: 10.1109/ICALIP.2016.7846617.

[16] A. Hogenboom, D. Bal, F. Frasincar, M. Bal, F. de Jong, and U. Kaymak, "Exploiting emoticons in sentiment analysis", in *Proceedings of the 28th annual ACM symposium on applied computing*, 2013, pp. 703–710.

[17] J. Perkins, *Python text processing with NLTK 2.0 cookbook*. Packt Publishing Ltd, 2010.

[18] R. Xia, F. Xu, C. Zong, Q. Li, Y. Qi, and T. Li, "Dual sentiment analysis: Considering two sides of one review", *IEEE transactions on knowledge and data engineering*, vol. 27, no. 8, pp. 2120–2133, 2015.

[19] A. Giachanou and F. Crestani, "Like it or not: A survey of twitter sentiment analysis methods", *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–41, 2016.

[20] E. Boiy and M.-F. Moens, "A machine learning approach to sentiment analysis in multilingual web texts", *Information retrieval*, vol. 12, no. 5, pp. 526–558, 2009.

[21] O. Kolchyna, T. T. Souza, P. Treleaven, and T. Aste, "Twitter sentiment analysis: Lexicon method, machine learning method and their combination", *arXiv preprint arXiv:1507.00955*, 2015.

[22] C. Cortes and V. Vapnik, "Support-vector networks", *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[23] V. Narayanan, I. Arora, and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model", in *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, 2013, pp. 194–201.

[24] T. Mitchell, “Machine learning, mcgraw-hill higher education”, *New York*, 1997.

[25] Z. Hailong, G. Wenyan, and J. Bo, “Machine learning and lexicon based methods for sentiment classification: A survey”, in *2014 11th Web Information System and Application Conference*, IEEE, 2014, pp. 262–265.

[26] *A twitter sentiment analysis tool.* [Online]. Available: http://www.sentiment140.com/.

[27] D. Hankamer and D. Liedtka, “Twitter sentiment analysis with emojis”,

[28] D. Kocich, “Multilingual sentiment mapping using twitter, open source tools, and dictionary based machine translation approach”, in *Proceedings of GIS Ostrava*, Springer, 2017, pp. 223–238.

[29] *Ontwikkelingen inkomend toerisme nederland.* [Online]. Available: https://nbtcmagazine.maglr.com/nl_NL/18311/253915/ontwikkelingen_inkomend_toerisme_nederland.html.

[30] M. Aiken and Z. Wong, “An updated evaluation of google translate accuracy”, *Studies in linguistics and literature*, vol. 3, no. 3, pp. 253–260, 2019.

[31] V. X. Gong, W. Daamen, A. Bozzon, and S. P. Hoogendoorn, “Estimate sentiment of crowds from social media during city events”, *Transportation research record*, vol. 2673, no. 11, pp. 836–850, 2019.

[32] S. Mohammad, M. Salameh, and S. Kiritchenko, “Sentiment lexicons for arabic social media”, in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, 2016, pp. 33–37.

[33] H. M. Pan, *How bleu measures translation and why it matters*, Nov. 2016. [Online]. Available: https://slator.com/technology/how-bleu-measures-translation-and-why-it-matters/.

[34] G. Levi and T. Hassner, *Age and gender classification using convolutional neural networks*, https://talhassner.github.io/home/projects/cnn_agegender/CVPR2015_CNN_AgeGenderEstimation.pdf, (Accessed on 28/04/2020).

[35] J. Clement, *Percentage of u.s. adults who use twitter as of february 2019, by age group*, https://www.statista.com/statistics/265647/share-of-us-internet-users-who-use-twitter-by-age-group/, (Accessed on 29/04/2020).

[36] A. A. Morgan-Lopez, A. E. Kim, R. F. Chew, and P. Ruddle, “Predicting age groups of twitter users based on language and metadata features”, *PloS one*, vol. 12, no. 8, 2017.

[37] S. Norena, *Finding correlation between many variables (multidimensional dataset) with python*, Nov. 2018. [Online]. Available: https://medium.com/@sebastiannorena/finding-correlation-between-many-variables-multidimensional-dataset-with-python-5deb3f39ffb3.

[38] E. D. Brown, “Will twitter make you a better investor? a look at sentiment, user reputation and their effect on the stock market”, *Proc. of SAIS*, vol. 7, 2012.

[39] S. Chatterjee, *Data correlation can make or break your machine learning project*, Oct. 2018. [Online]. Available: https://towardsdatascience.com/data-correlation-can-make-or-break-your-machine-learning-project-82ee11039cc9.

[40] *Spearman’s rank-order correlation.* [Online]. Available: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php.

[41] A. Ganti, *Correlation coefficient definition*, Feb. 2020. [Online]. Available: https://www.investopedia.com/terms/c/correlationcoefficient.asp.

[42] *Pearson product-moment correlation.* [Online]. Available: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php.

[43] C. Croux and C. Dehon, “Influence functions of the spearman and kendall correlation measures”, *Statistical methods & applications*, vol. 19, no. 4, pp. 497–515, 2010.

[44] J. Magiya, *Kendall rank correlation explained.* Nov. 2019. [Online]. Available: https://towardsdatascience.com/kendall-rank-correlation-explained-dee01d99c535.

[45] R. Morey, *Shared vs VPS vs Dedicated vs Cloud Hosting*, https://wp-rocket.me/blog/shared-vs-vps-vs-dedicated-vs-cloud-hosting/, (Accessed on 01/05/2020),

[46] L. Dignan, *Top cloud providers in 2020: AWS, Microsoft Azure, and Google Cloud, hybrid, SaaS players | ZDNet*, https://www.zdnet.com/article/the-top-cloud-providers-of-2020-aws-microsoft-azure-google-cloud-hybrid-saas/, (Accessed on 01/05/2020),

[47] Apache Kafka, *Introduction | Apache Kafka*, https://kafka.apache.org/intro, (Accessed on 01/05/2020), 2020.

[48] J. Kreps, *It's okay to store data in kafka*, https://www.confluent.io/blog/okay-store-data-apache-kafka/, (Accessed on 01/05/2020),

[49] M. Seifert, *Building Systems with Transactions Using Apache Kafka | Confluent*, https://www.confluent.io/blog/transactional-systems-with-apache-kafka/, (Accessed on 01/05/2020),

[50] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual analytics: Definition, process, and challenges", in *Information visualization*, Springer, 2008, pp. 154–175.

[51] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations", in *Proceedings 1996 IEEE symposium on visual languages*, IEEE, 1996, pp. 336–343.

[52] TU Delft, *Urbanism.* [Online]. Available: https://www.tudelft.nl/bk/over-faculteit/afdelingen/urbanism/, (Accessed on 22/04/2020).

[53] ProductPlan, *Moscow prioritization.* [Online]. Available: https://www.productplan.com/glossary/moscow-prioritization/, (Accessed on 22/04/2020).

[54] Scrum.org, *What is scrum?* [Online]. Available: https://www.scrum.org/resources/what-is-scrum, (Accessed on 22/04/2020).