Bende Gij een Brabander?

Group Design Report

Designing with Advanced AI

Eva van der Born / Jordy Alblas / Almar van der Stappen / Lei Nelissen

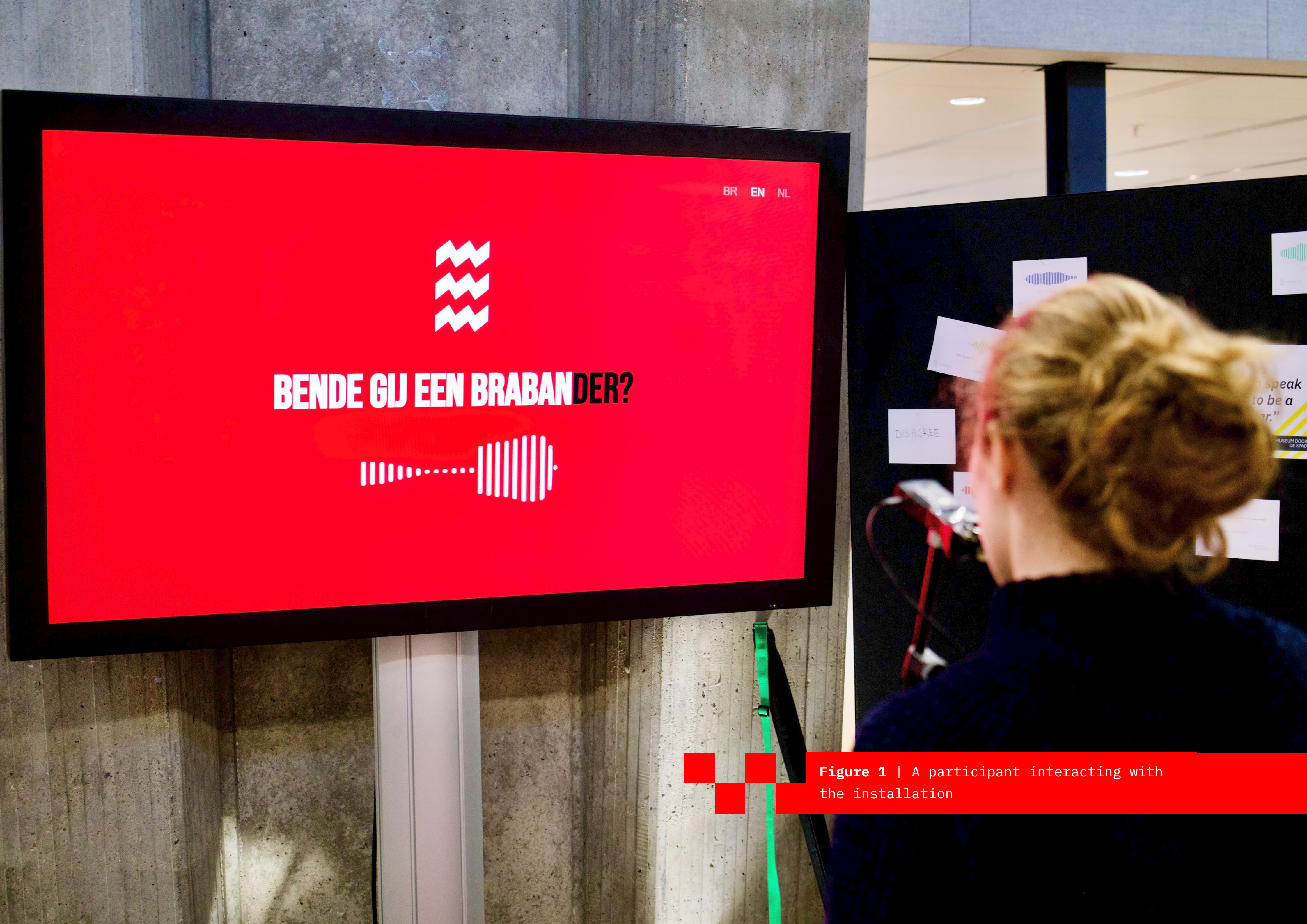


Figure 1 | A participant interacting with the installation

Abstract

The system delivered as described in this study is the result of a design exploration that serves to showcase the ability of AI as a facilitator of personalized experiences in a public context. The specific use case provided was that of designing for the Eindhoven-based “Museum door de Stad”, whose ambition it is to entertain visitors with a live interpretation of personal data through AI, in the context of celebrating a hundred years of Eindhoven.

The proposed design is an installation which focuses on the pronunciation of a specific sentence in the Brabant dialect: “Bende gij een Brabander?”. A pre-trained model is able to distinguish whether the accent in the sample is Brabants or not. This factor is then used to provoke discussion on what it means to be a Brabander, by means of theses, with which a participant can agree or disagree by pinning a printout of their sample to a wall.

Through these discussions, we engage participants to look more critically at what it means to be a Brabander, and how speech and accents factor into this identity.

Eindhoven, 26-01-2020

Background

Museum door de Stad

Museum door de Stad (Dutch for Museum throughout the City) is an initiative by the Eindhoven Museum, a museum that focuses on exploring the rich cultural and technological history of Eindhoven through direct interaction with the public, sparking debate and reflection [24]. In this regard, the Museum door de Stad initiative focuses on using interactive installations as a means of provoking debate on a number of topics, among which freedom was the first one. Using data is an explicit part of the designed experiences [25].

In the newest iteration of this initiative, the celebration of the creation of modern-day Eindhoven, a hundred years ago, is the primary inspiration of investigating how Eindhoven came to be, in particular through its city districts. Themes of identity are again explored through interactive installations and active participation by the public at large [26]. Particularly this latest exhibition is the specific target for the design as detailed in this report.

Socio-Cultural Context

As a client, the Eindhoven Museum puts a spotlight on culture in the region of Eindhoven and how it has evolved over time. They have dedicated a lot of their expositions to showcasing the day to day life of Brabanders throughout time, a term used to describe inhabitants that represent or feel they truly represent the Southern Dutch province. Individuals that describe themselves as a Brabander generally find pride, a sense of community and inclusivity in the feeling of belonging to the region [23], but what does it truly mean to represent a region as a person?

Eindhoven, like many cities in the Netherlands, has seen a shift in demographic in the last 50 years. A growing technical industry has made the city attractive for expats and a large city center as well as a number of cultural factors have attracted immigration as a whole [21]. As the city demographic changes, what can be perceived as the ‘average’ citizen does as well. Although this process is ongoing, opinions on what it means to be a Brabander seem to not have changed over this period, resulting in a contrast between current and future inhabitants and the reflection of the region [19].

Feeling like you are part of the region that you are currently inhabiting can go a long way towards boosting satisfaction with living there [16]. Because of this, the topic of inclusivity is often mentioned when decisions are made regarding socio-economic initiatives [4,17]. However feelings of inclusion are subjective and can be hard to pin down. Getting a better grasp on what it means to feel included in a region could be a real benefactor to making decisions about the region in the future and as such we believe it should be considered a prime subject of discussion. We look to challenge ideas of what it means to be a Brabander in the current day and culture, as well as how that might change in the future.

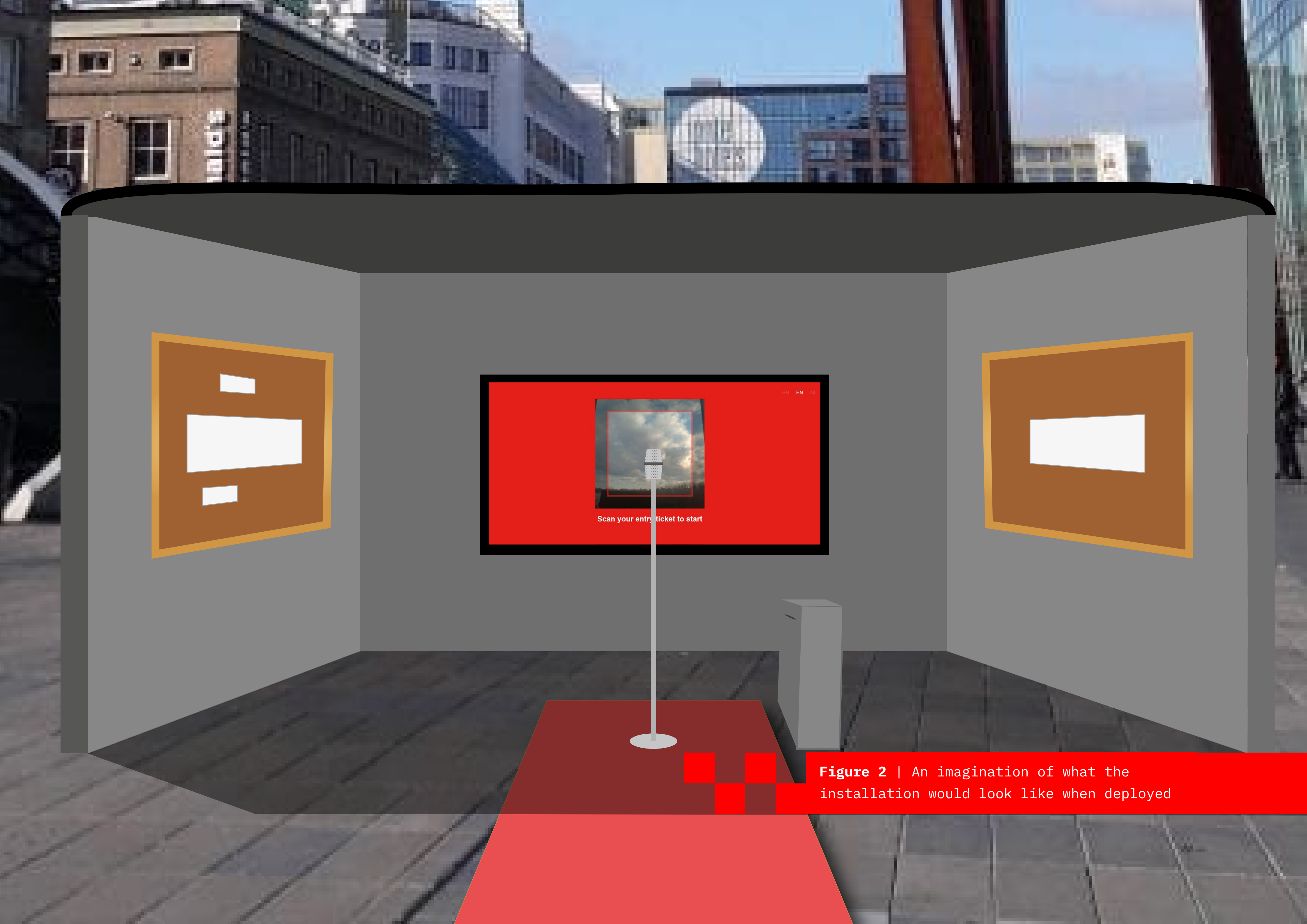


Figure 2 | An imagination of what the installation would look like when deployed

Conceptual Design

Mission and Means to Achievement

Given the context and situatedness of the design in the exhibition of the Eindhoven Museum, we would like to spark a discussion that involves the subject matter: "What does it mean to be a Brabander?" Through an interactive installation, we stimulate people to think about what it means to be part of and identify with the Brabant culture. To encourage interaction with the installation, located somewhere on the streets in Eindhoven, and enable these discussions, we aimed to build something that is fun, accessible and fast to use by visitors during their visit to the city centre of Eindhoven.

Group Motivation

At the start of this course, we discussed and formed our group around what we wanted to achieve and learn in these months. We defined our personal motivation and all shared a couple of similar goals. We wanted to dive deeper into the technical aspects of an AI-powered interactive installation. Most of us have a background in web-based design and development and we wanted to improve our understanding of how to empower these designs through AI.

Therefore, we decided to come up with the conceptual design as described. We built a web-based interface using the language that most of us used before (JavaScript) and connected this to a Python back-end. By building a voice classification specifically, it enabled us to work on Machine Learning at a greater extent than what we were used to do and it helped us to improve our (technical) understanding. The overall concept as described helped us to combine both our skills and achieve our personal goals, and it made us develop a functional prototype with a connected front- and back-end.

Installation

Decorated with big, remarkable Brabant signs, the installation attracts people that are passing by. A red carpet and a big red screen invite visitors to come closer and perhaps start an interaction. Interacting with the installation allows users to test part of their Brabantian origins by evaluating to what extent their speech is considered to fit the Brabant dialect. If one is interested in finding out how they would score, all they would have to do is walk up the red carpet, where a microphone remains, next to a small elevated box.

Participants are asked to speak a predetermined sentence into the microphone, while the screen shows an animation with the analysis of the recording. As soon as a verdict is reached, the percentage match with the dialect is shown. If it exceeds a certain value (80% in this case), the person is declared to be a Brabander, after which music plays and confetti explodes. If the value is not exceeded, the participant can try again and a motivational text is shown.

At the same time, a small A6 card is printed from the little box showing an audio visualization of the recording together with the dialect score in percentage. Finally, on both sides of the installation, a statement about Brabant is presented where users can stick their card along a decision gradient (from left [do not agree] to right [do agree]). These statements change every couple of days and all relate to being or feeling a Brabander. For example:

“As a Brabander, you have to celebrate carnaval each year.”

“You have to speak Brabants to be a Brabander.”

“I am a Brabander”

“To be a Brabander, you have to live in Brabant for at least years.”

“A real Brabander is born in Brabant.”

“I can recognize a Brabander just by looking at them.”

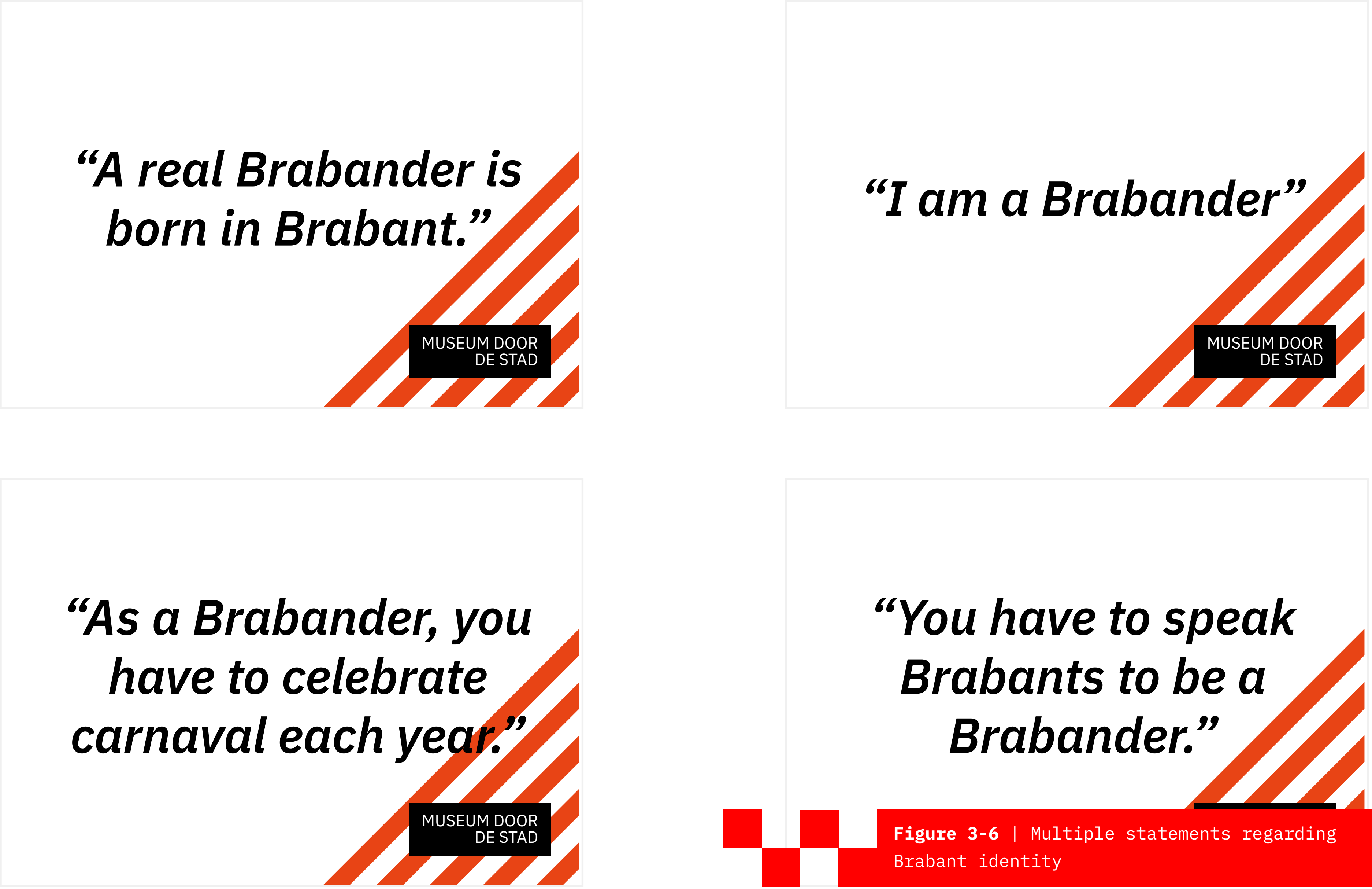
The participant is engaged by a presenter, dressed in full Brabant clothing, so that he or she can explain their opinion. If the presenter is not available, visitors can, of course, still stick their cards to the board. Potentially, a screen can replace the presenter and ask a couple of follow-up questions instead.

This entire interaction should spark a discussion on identity and inclusivity in the regional society of Eindhoven, whilst also generating ideas and visions on how the concept of ‘being a Brabander’ evolves in the future. Input can eventually be used by the municipality to improve the bonding of citizens with Eindhoven.

Let’s say that a certain visitor visits the exhibition on a Friday, interacts with the installation, and gains a score of 90%. The statement on the board louds: “As a Brabander, you have to celebrate carnaval each year.” If the visitor does not agree with the statement, (s)he can stick the card at a position on the left side of the decision gradient where (s)he feels comfortable with. If it appears that many other people with high scores do not agree, this provides insights in the importance of celebrating carnaval for being a Brabander among Brabanders. On the other hand, another visitor could visit the exhibition and gain a score of 10% (for example, the visitor recently moved to Brabant from the north of the Netherlands), and the person fully disagrees with the statement “To be a Brabander, you have to live in Brabant for at least 15 years.” Based on conversations, important insights could be gathered on why this person felt at home so quickly.

Personalised Experience

The design will be situated in a pop-up exhibition that is organised by the Eindhoven Museum. This stand will feature a microphone, a screen and a presenter (preferably in typical clothing). The stand should be stylised in typical Eindhoven fashion, possibly featuring either Eindhoven carnival colors (blue and



“A real Brabander is born in Brabant.”

MUSEUM DOOR DE STAD

“I am a Brabander”

MUSEUM DOOR DE STAD

“As a Brabander, you have to celebrate carnaval each year.”

MUSEUM DOOR DE STAD

“You have to speak Brabants to be a Brabander.”

Figure 3-6 | Multiple statements regarding Brabant identity orange) or the Brabant checkered flag (red and white). Additionally, this serves to create an environment in which the (stereo)typical view of Eindhoven and Brabant as it is currently seen is exhibited.

Connection to Other Installations

The 'Museum through the City'-exhibition exists of many different installations across the city. We envision a personalized experience where installations interact with each other and learn from previous visits, both from the same and other visitors. Moreover, it recognizes visitors that already visited another installation in another part of the city. A great example of a similar experience is what De Efteling does with its talking tree: when a visitor passes by and has the Efteling app installed, the tree starts talking and mentioning the visitors name or telling a personalized, tailored story [27]. We see a similar experience for the Museum through the City: interaction with installation A does affect interaction with installation B. We built in the basics of such a functionality through a QR code scanner. Before the interaction starts, visitors need to scan their unique QR code. Data from the installation (such as the audio visualization or the percentage score) can be stored in Data Foundry as a unique record with the id of the visitor to be used by other installations.

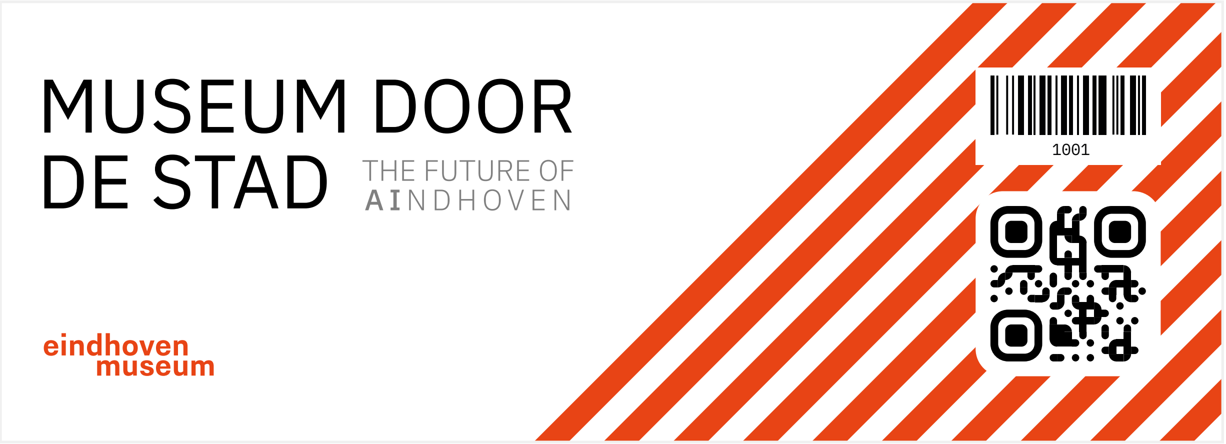


Figure 7 | The design for an exhibition entry ticket with QR Code

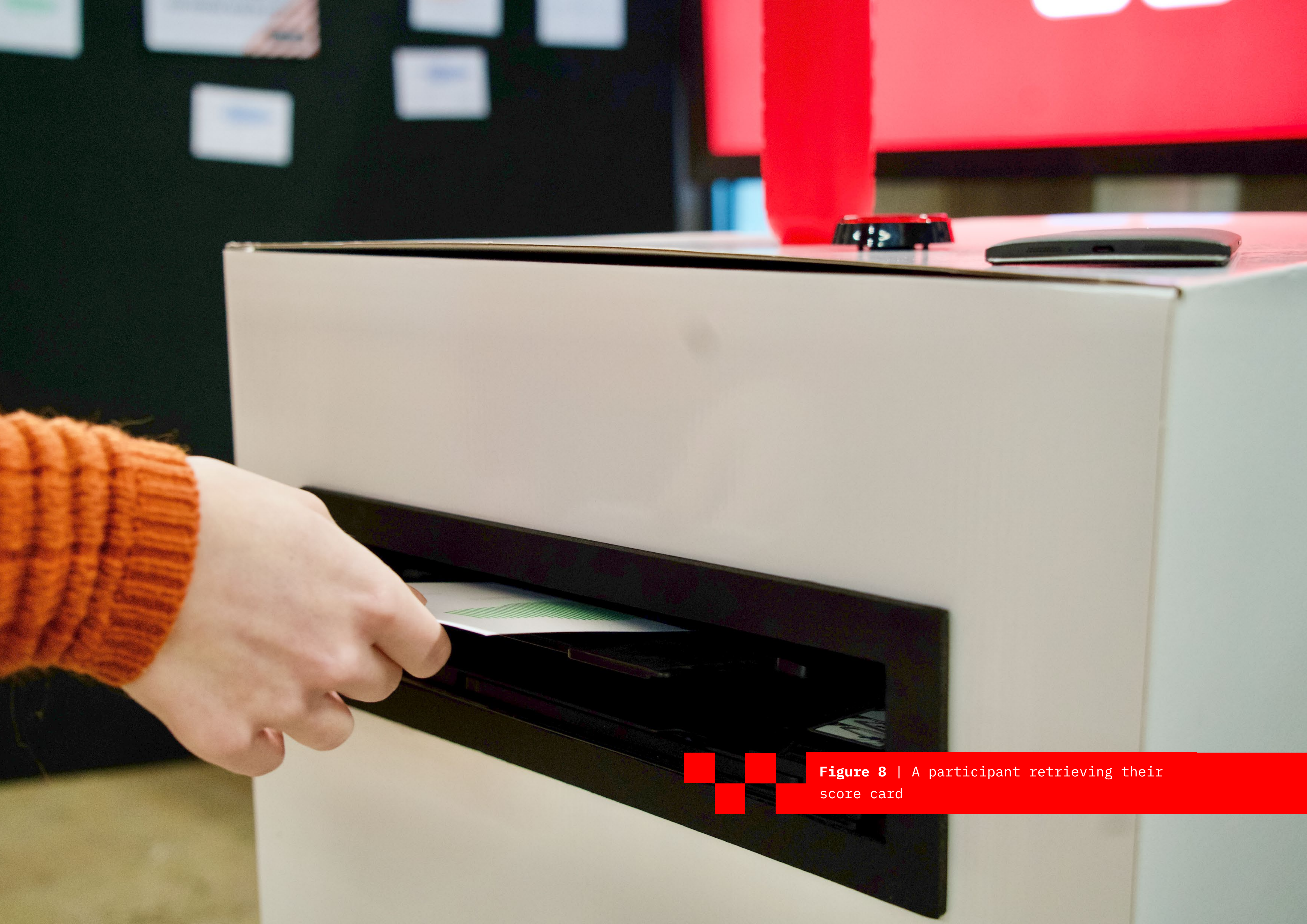


Figure 8 | A participant retrieving their score card

Figure 9 | The environment in which the samples were collected

Data Collection

Voice Data

As discussed in earlier sections, the envisioned prototype would use parameters of auditory data to determine to what extent a new user can be said to have a Brabant dialect. The Brabant dialect is a telltale sign and probably the most commonly referred to factor in determining that someone is a Brabander. Specifically, this dialect can be distinguished by word usage and a couple of pronunciation differences, for example its soft pronunciation of G-tones [11].

Although there is room to use a ton of other features to classify whether or not someone is a local, the voice classification is somewhat unique in its prominence for that distinction as well as its unique data type. With a unique data type we look to refer to the fact that this auditory feature could not be obtained from any of the other installations that were being created for the Eindhoven Museum. As such, this would allow for future expansion of the system, incorporating the data gathered by other installations as additional features for its prediction model.

Feasibility

When considering the initial viability of the dialect classification, a number of solutions were benchmarked that had used machine learning to do such classifications in the past. The most notable of these were able to achieve close to 90% prediction accuracy [9,14]. In these papers, researchers were able to explore audio represented through vector machines [5,9,12] as well as MFCCs [3,7], which in turn could be used as input for a learning model. This initial benchmark served as our proof of concept that such a task would be possible at all, and the technical challenge as such would be to emulate similarly significant results using those methods.

Voice Parameters

There are a number of auditory parameters that could potentially be a tool for classification of voice recorded audio samples. Some examples are:

Using tonal emphasis or harshness makes it so that there is a large deviation from the average sound level across an audio fragment. Looking at the sound pattern, this is easily distinguished by relatively large spikes throughout. This can also vary depending on what was spoken, but can serve as a means to identifying different pronunciations if the spoken text remains unchanged between samples. For example, with the Brabant dialect, the expectation would be that if a sentence ended on a word including the letter G, the audio sample would express a lower tonal harshness compared to the traditional Dutch language.

Silence between words or surrounding certain tones can also be a good indicator of sound. Although complete silence is almost never seen in an audio recording, we will refer to silence as simply non-speaking. For many programs the threshold of non speaking is considered to be anything below 20% of the average sound level throughout a recording.

If a sample is not normalized classifications can also be done based on what the average sound level, or sound power (P) is. A higher sound power throughout a sample means louder speaking, which can be used to differentiate between different contexts.

For any audio sample the words spoken will obviously play a role in its similarity to other samples. If one were to differentiate classes based on what text is spoken, one could look at a normalized sample, where the places of silences compared to those of high sound power are the main feature to be considered.

Speed

A fast speaker is easily distinguished from a slow speaker by humans and the same can be done by a system. If there is a predefined text to be spoken, the length of the sample can be a prime feature, but for undefined words, the length of individual silences between words would likely be the main determinant.

Pitch

Pitch can be a good indicator of a number of things, and is easily determined by the frequency of a sound sample. High pitched notes carry a high frequency whereas lower ones do the opposite. Although being very prone to error, pitch can generally serve as a feature to classify gender or adulthood with slightly better accuracy than achieved randomly.

Design Perspective

As designers, having all these different parameters to work with going into training a model, this is not only a technical challenge but also a design challenge. When building a system like this, a designer can direct and control the data input that is fed into the model, while consciously excluding other parameters. For our system, some of these decisions were made on the basis that dialect nuances are more difficult to forge compared to other instances, like words spoken or overall sound level, while others were excluded simply because of the context.

To be more specific, it was decided that the auditory samples would be collected in such a way that the emphasis would be on tonal differences between samples. As such the model would only predict if a user spoke the local dialect or not, and the experience (data collection phase) was tailored to that specifically. This way of collecting data would highlight the differences that often come forth in local dialects or accents by excluding the other parameters, that are more easily misdirected. Whereas a fake accent can be attempted, this is generally much harder to do than faking other parameters. One could think of, for instance, word usage. If words could differ individually, it would not only be extremely difficult to build an accurate model, it would also make it so that one could easily learn the ‘passwords’ that lead to a high accuracy prediction of Being a Brabander. Similarly, loudness is easily mimicked, but also unreliable for a different reason. Since this installation is supposed to be featured in the middle of the city, it should be considered that loudness or silence outside of speech are parameters that are quickly corrupted by outside influences. Individual differences between samples could thus be caused simply by vehicles or people making noise around the installation, which is why we believe that these unnormalized parameters should generally be considered as bad predictive features.

Collecting

As described, the goal for data collection was to collect samples in such a way that would highlight the tonal differences between samples, by limiting the other differentiating features present in the samples. It was thus decided that samples should feature the same words and should be isolated to those words actually being spoken. For this specific purpose the design of the system was altered to provide users with a general speaking tempo and sentence to speak.

The sentence for users to speak was chosen to be “Bende gij een Brabander?”, which translated from the Brabant dialect would be “Are you a Brabander?”. The reason for choosing these words is that they include multiple instances of the sounds that characterize the dialect, such as the pronunciation of G’s and R’s, while also being a little bit tongue-in-cheek in respect to the goal of the installation.

To actually collect the samples, two groups of people were approached to speak the mentioned sentence into an audio recorder. The first group consisted of students on the Eindhoven University campus. The second group consisted of city centre shoppers. Both were approached with a microphone and asked to speak the sentence in their normal mannerisms. To properly support both sides of the dataset, it was attempted to match the number of non Brabander samples roughly evenly with the number of Brabander samples. The total number of samples collected came out to be 69 recordings.

Data Retrieval

As for the data that was used in this study of dialect, the audio samples can all be retrieved from public (or at least university-accessible) sources. All samples were intended to be submitted via Data Foundry, but unfortunately it does not accept any upload requests larger than one megabyte. Thus, the original files will be submitted with the Canvas report instead. All testing and training data as input for the model can also be found in the public repository (Appendix A) under ‘/mfcc/test_dataset’ and ‘/mfcc/training_validation_dataset’ [10]. The original samples include all available metadata of their original recording and can be freely downloaded from the above sources.

Alternative Approaches

Having decided on this approach to utilizing and collecting data, it is still worth reflecting on alternative design decisions that could have been made to support the desired experience. In questioning what it means to be a Brabander, we had made the decision to focus on a singular feature, being the tonal differences that represent the spoken dialect. Although there is an indication that the dialect covers the tonal differences on specific letters as well as the vocabulary [18,28] there is no indication that it is limited to such parameters. It could, for instance, be the case that spoken differences also manifest themselves in generally louder speaking or that Brabant locals have adapted to speaking a lower pitch on average. When using a learning solution like a neural net, over time the model should be able to learn these differences and thus grow towards using the proper parameters. With an enormous sample size, this would thus seem like the proper implementation, as the relevant parameters would be isolated and the model would be expected to grow very accurately. However, because we were forced to work with a limited sample size in this project (being the limit to how many samples we were able to collect), this large number of differences between samples would most certainly result in the model overfitting for the given inputs. With this in mind, the focus was put on isolating a single voice parameter and trying to predict for that one as effectively as possible, but if the project were to be run again with a larger sample size, inputting more raw and individually different data could be considered.

Datamining

First Iteration

Since it took us time to collect all the data required for the model, we already started to build the pipeline with an open-source accent data set named ‘Speech Accent Dataset’ from the Kaggle website [30]. The dataset contains parallel speech samples from 177 different countries. These samples are recorded audio fragments of participants reading the same sentence in English. To make an indication whether it was actually possible to do a decent classification on two accents, we used this dataset and picked speech fragments from native speakers from England and native speakers from the Netherlands. This resulted in a dataset that consisted of 50 speech samples from native speakers from England, and 47 speech samples from native speakers from the Netherlands, whom all pronounced the same sentence in English.

The steps described below have all first been executed with this subset of the open-source dataset with Kaggle. Several things were learned from this first iteration, for example how we had to preprocess the data and how the parameters of each model had an influence on the final result. The parameters from the machine learning and deep learning models we used were first adjusted to this specific dataset, and therefore needed some re-adjusting before it was compatible with our dataset. Parallel to this first iteration, we started with the process of collecting our own voice data and preprocessed this according to the things we learned in the first iteration.

Preprocessing

Filtering the Data

Our collected voice data samples sometimes contained recordings that were not related to the one sentence our participants had to pronounce. By listening to all the samples again, the outliers (for example a data point wherein we were talking nonsense together) were manually removed. We removed a total of 13 data points that were classified as outliers.

Validation with a Native Brabant Speaker

The next important thing to do was to assign the class value ‘Brabants’ or ‘Non-Brabants’ to a specific datapoint. 22 year old student Jolie Smets, born and raised in Helmond (Noord-Brabant), provided external validation on the total of 69 data points by listening to them and making the division between Brabanders and Non-Brabanders, based upon their accent. Ultimately, this resulted in a division of 31 data points with the class ‘Brabants’ and 38 data points with the class ‘Non-Brabants’.

Noise Removal + Normalization

Since the audio samples were recorded outside, some background noise was present. Moreover, the wind also affected some samples quite heavily. Lastly, the recorder gain was unintentionally adjusted during recording, which caused differences in loudness between samples. In order to resolve these issues, all samples were imported into Adobe Audition for further processing. First, the Sound Remover process was used to eliminate the wind noise from the samples. All samples were then normalised. These samples were then used directly for training.

Splitting Dataset into Subsets

After the noise was removed and normalization was applied, it was necessary to split the dataset, that in total consisted of 69 datapoints, in two subsets: a training/validation set and a test set. An overview of this can be found in Figure 10.

Training and Validation

The training/validation set, containing 80% of the dataset (57 data points), was specifically used for training and optimizing the different models. This subset was separated from the test set by randomly picking 57 data points from the overall dataset and then placing it into a separate folder. This subset was again split (by using the sklearn.model_selection.train_test_split function) into a training subset and a validation subset, using a 80/20 percentage split in each model. The 80/20 percentage split is a common rule of thumb, most often used to split the data [29]. The validation training set is important for tweaking detailed parameters and reducing overfitting.

Testing

The test set, containing the other 20% of the dataset (12 data points), was specifically used for evaluating every single model. This data was left over as a result of randomly picking training data and was also placed in a separate folder. The 12 data points were split with a 50/50% division, resulting in 6 test data points with the class 'Brabants' and 6 data points with the class 'Non-Brabants'. The test data has not been used during the training process and has only been used after the training and validation phase ended, thus resulting in an unbiased model. The testing phase is important to demonstrate

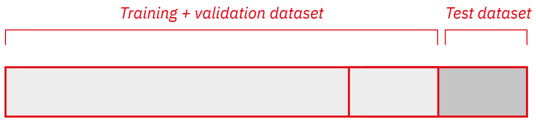


Figure 10 | The dataset was split into three parts

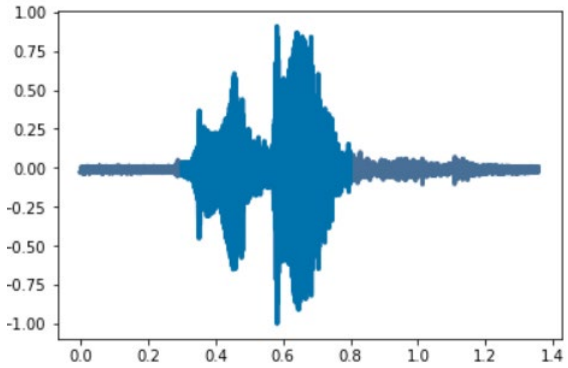


Figure 11 | An example of how the audio is cut by the splicer algorithm

the reliability of the model and how well the model performs.

Audio Slicing

The first thing that has to be done to surpass large individual differences between speech samples is to filter out the differing amounts of silence. In our model, we only want to account for the actual speaking portion of audio samples, rather than classification being based on how long people wait to speak or how much noise is in the background.

For this reason we built a splicer to splice the silent segments out of the audio inputs, which was an adapted version of the audio slicing benchmarked by a related study [8]. The splicer takes all the .wav format audio files in a designated “/Sounds” directory and delivers a version for each audio file with the silence cut out respectively. These are then saved to another designated directory called “/Soundsimproved”.

The ThinkDSP library is an audio management library that allows one to read, draw, and edit waveform audio signals in Python [6]. This library was used to cut the actual silence out and to calculate the wave amplitude across time for different intervals. An example of a soundwave and the cut made by the slicing algorithm can be seen below, with the differences in blue indicating the changes made.

The algorithm was written as such that it splits up the audio in a number of separate pieces and then calculates the average amplitude over time for every single one. If the average for a segment comes out to be less than 20% of the average amplitude across the entire sample, that part of the audio is determined to be silent. For the revised sample, every part up to the first non-silent one is cut, as well as every single part past the last non-silent one.

Although the slicing algorithm was built for the envisioned pipeline, it is currently not implemented in the actual live interaction. It can, however, be used for the training and test set. To finalize this aspect of the prototype, one would have to call the script in between recording and classifying the sample.

MFCC Extraction

To make the .wav files usable for any kind of machine learning methods, they had to be pre-processed into numeric values. MFCC, short for Mel-Frequency Cepstral Coefficients, is a feature often used in audio and speech recognition. As can be seen in Figure 12, the process of extracting MFCC is a highly mathematical process. We used the *librosa* library, a python package for music and audio analysis, to do this. A function for MFCC extraction for a given audio file was defined. The underlying process was analyzed and, based upon this, our number of MFCC bands was set. When looking at the duration of each sound sample, a total of 120 MFCC bands seemed applicable. The example from the paper used a total amount of 40 MFCC bands. Nevertheless, the duration of our audio sample was 3 extra times the size of the duration of audio samples in the example. So we 3-doubled the number of MFCC bands, from 40 to 120.

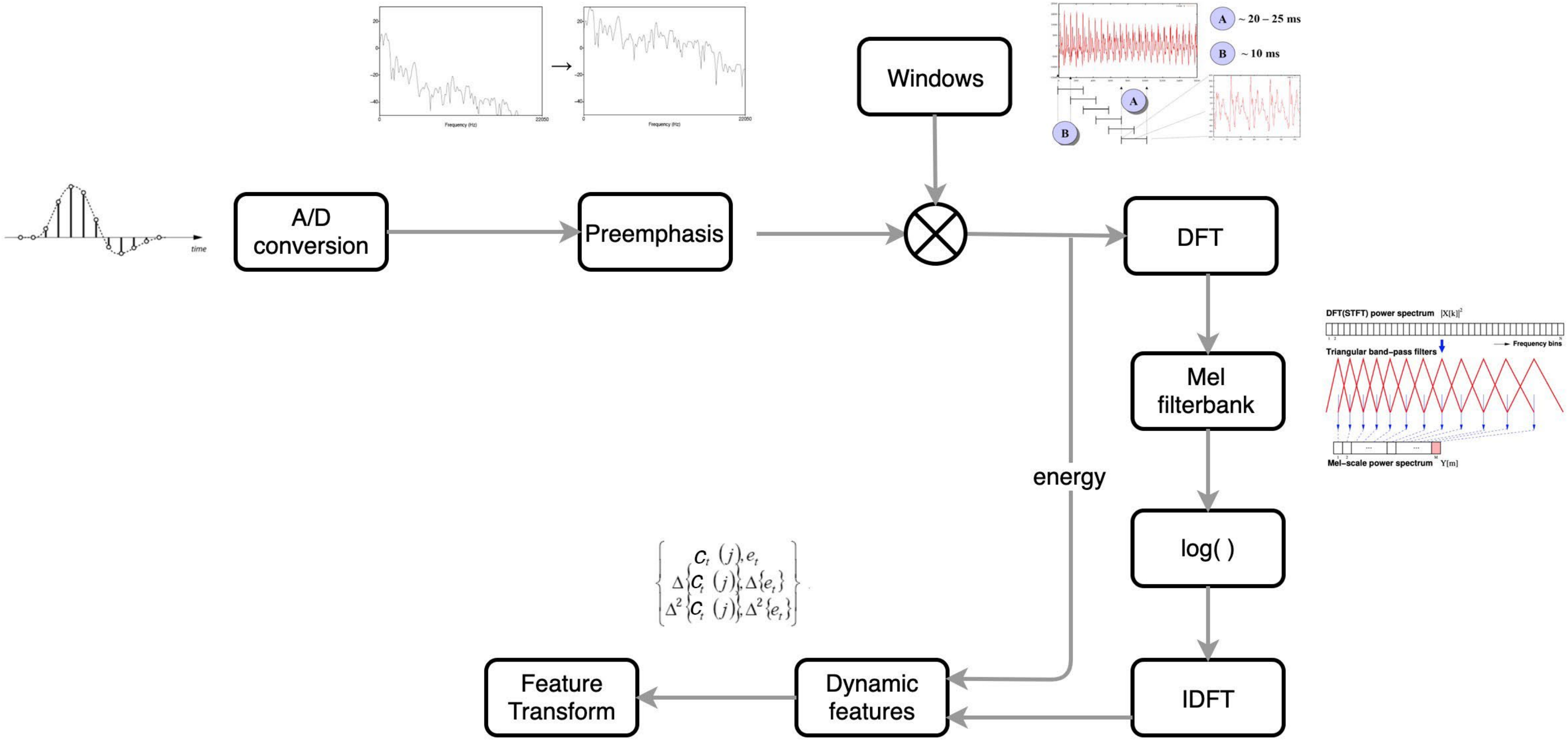


Figure 12 | An overview of the data flow in the installation application

After extracting the MFCC bands for both the Brabant and Non-Brabant data, these values were put in a separate feature array and were classified with their parent classifier. Lastly, the resulting array was put into a pandas dataframe and, through the use of pickle, saved and stored in the MFCC-folder. Running the code once was enough to do the job. This was both done for the training & validation dataset and the test dataset. The final script can be found in the DWAAI Github folder under the name 'mfcc_extraction_brabant_nonbrabant_training.py'.

Learning

Training

In work done by Sheng & Edmund [14], two traditional machine learning methods (gradient boosting and random forest) and two deep neural network architectures (1D Convolutional NN and Multilayer Perceptron) were proposed and tested on their accuracy. In their paper, *Deep Learning Approach to Accent Classification*, accent classification was made based upon an English sentence that was pronounced by native English speakers and speakers from Asia. The traditional machine learning models showed an accuracy of 69% or higher and the deep learning models showed an even higher percentage of 81%. Therefore we decided to pursue their direction and created three different models: one based upon gradient boosting, one based upon random forest, and one based upon a one-dimensional convolutional network.

The first two classifiers that we tried to implement were the ensemble learning methods Random Forest and Gradient Boosting. Both methods were relatively easy to implement through the use of *sklearn* [13], an open-source library for implementing machine learning in Python.

Random Forest is a supervised learning algorithm that fits a number of decision tree classifiers and randomly picks data samples and gets a prediction on each of these samples. The decision trees together form a so-called 'forest' [22,31]. It uses averaging to improve its accuracy and to reduce overfitting. It also uses voting to get the best prediction for each decision tree. We experimented with a few parameters of the algorithm in order to tune the algorithm. After minor tweaking, the parameters were set to the following values: n_estimators (number of decision trees) was set at 10 because our dataset was relatively small, and random_state (randomness) and the max_features (maximum of features) were set at 120 because our number of features that was extracted from the MFCCs was 120. By setting these values, 5 classes with the value 'Brabants', and 6 classes with the value Non-Brabants were picked as validation subset, and thus resulted in an almost equal division between the two classes. As a result that was seen in the classification report, the accuracy of the model varied between 0.73 and 0.81. The final script can be found in the DWAAI Github folder under the name 'random_forest.py' [10].

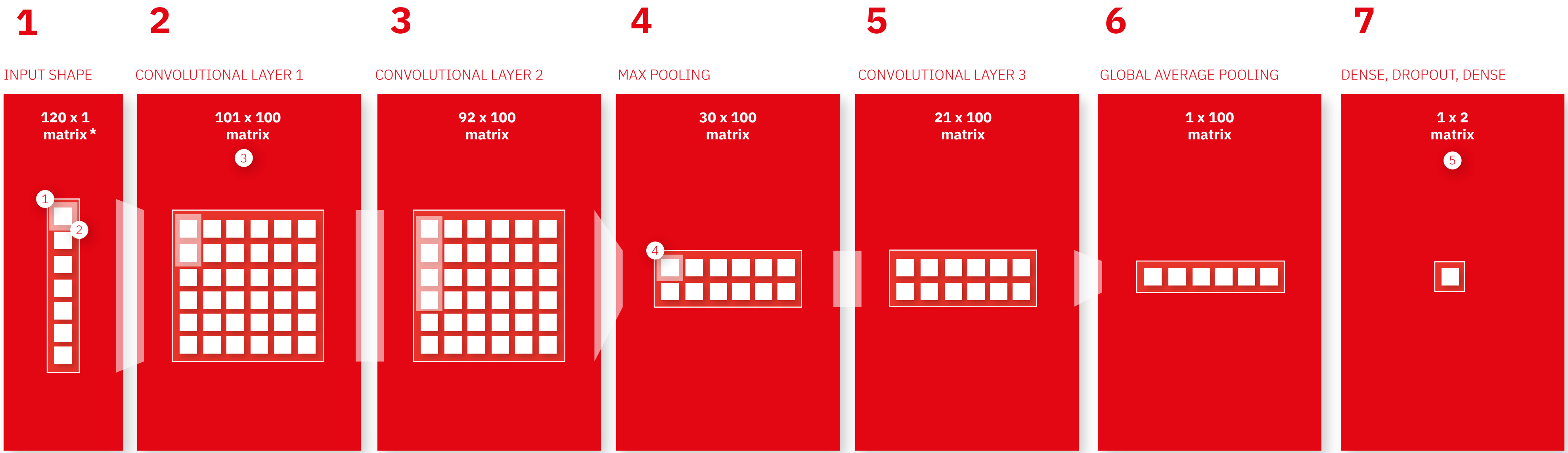
Gradient Boosting is another supervised learning algorithm wherein a technique named ‘boosting’ is applied [15]. Boosting is a process of improving and tweaking the weight of the features in every creation of a new tree, thus resulting in a model wherein each new tree is a new fit on improved version of the original dataset. Specific for Gradient Boosting is that this method uses the gradients in the loss function to indicate the performance of the coefficients for fitting the data. We experimented with a few parameters of the algorithm to tune the algorithm. After minor tweaking, the following parameters were set: n_estimators (number of decision trees) was set at 20 because our dataset was relatively small but the value of 10 (as used in random forest) provided a lower accuracy, random_state (randomness) was set to 10, and the max_features (maximum of features) was set at 120 because our number of features that was extracted from the MFCCs was 120. Also, we played with the learning rate of the model. By giving a list of possible learning rates for the model and printing a classification report for each learning rate, it was easy to compare the influence of the learning rate on the model. As a result of this, a learning rate of 0.5 provided the highest accuracy; 0.63. By setting these values, 6 classes with the value ‘Brabants’, and 5 classes with the value Non-Brabants were picked as validation subset, which thus resulted in an almost equal division between the two classes. By printing the classification report, our model resulted in an accuracy of 0.63 with a learning rate 0.5. The final script can be found in our GitHub repository under the name ‘gradient_boosting.py’ [10].

We wanted to challenge ourselves to learn the basics of neural network architectures. Therefore, we created our final and third model: a 1-Dimensional Convolutional Neural Network that uses deep learning methods to build a neural net with the help of Keras, a Python deep learning library.

One-, two-, or three-dimensional convolutional neural networks are common deep learning methods for applications such as speech and image recognition [1,2]. It also applies significantly well to the analysis of audio signals, since the data is signal data over a fixed-length period of time. We chose to create a 1-Dimensional Convolutional Neural Network because our dataset is one-dimensional. After importing our saved pandas dataframes into the 1D-CNN by using the pickle module, our features and its corresponding classification labels were converted into numpy arrays. It was important to hot-encode our classification labels, resulting in the class label 0 for ‘Brabants’ and a class label 1 for ‘Non-Brabants’. The data was split with a 80/20 split into a training and validation subset. Before our data enters the first layer of the model, it required a reshape into a three-dimensional array. This is necessary because since we had to split the data on samples, frequency bands, and channels.

The first layer of the model, a Sequential layer, receives information about the input shape of our model, which in our case was (42, 120, 1). After going through the Sequential layer, it enters a Gaussian Noise layer wherein noise is added to increase our sam-

1D-Convolutional Neural Network



1 INPUT SHAPE
120 x 1 matrix*

2 CONVOLUTIONAL LAYER 1
101 x 100 matrix

3 CONVOLUTIONAL LAYER 2
92 x 100 matrix

4 MAX POOLING
30 x 100 matrix

5 CONVOLUTIONAL LAYER 3
21 x 100 matrix

6 GLOBAL AVERAGE POOLING
1 x 100 matrix

7 DENSE, DROPOUT, DENSE
1 x 2 matrix

HEIGTH
This is the length of the dataset. In our case this is 120, since every array consists of 120 MFFC bands.
WIDTH / DEPTH
The width of the dataset, which in our case is 1.

1. FILTER LENGTH / KERNEL SIZE
The sliding window with the size of 10 (or 20 in the second Conv layer), tht will slice through the data.
2. FILTER / FEATURE DETECTOR
Our dataset has 100 feature detectors and a kernel size of 10. The window will slice through the data for 111 steps.

3. OUTPUT
Output after the first Convolutional Layer.

4. POOLING
Pooling will reduce the output of the layer in order to prevent overfitting. We use max and global average pooling.

5. FULLY CONNECTED (DENSE, DROPOUT, DENSE WITH SOFTMAX)
The last few layers aim to reduce overftting as well and prepare the data the predict the probability for the 2 classes through the use of the softmax activation in the last dense layer.

* The graphic of the matrix deviates from the actual values

Figure 13 | The layer setup for the convolutional neural network ple size. Within every datapoint, one MFCC value is stored. Since we have 120 numbers of MFCC bands in each datapoint, this results in a 120 * 1 matrix, wherein 120 is the height and 1 is the width.

After it has entered the Gaussian Noise layer, the data enters the first Convolutional Layer. This layer has 100 feature detectors and a kernel size of 20, therefore the window will slice through the data for 111 steps (120+1-20). Another Convolutional layer was added afterwards. Having a second layer prior pooling with a filter of 150 and a kernel size of 10 will result in a matrix of 92 x 150 (101+1-10), and allows the model to learn more features.

We use standard max pooling to get ⅓ the output of the previous layer and prevent overfitting. The output of this layer is only a third of the input layer, resulting in a matrix of 30 x 150. Afterwards, the data will pass the third Convolutional layer with kernel size 10 will result in a matrix of 21 x 100 (30+1-10), in order to learn more high level features.

The Convolutional layers got followed by a last pooling layer: a Global Average Pooling layer. Global Average Pooling takes the average output of each feature from the previous Convolutional layer, and thus reduces the size of the data and prepares the model for the final layers. This results in a matrix with the shape of 1 x 100.

The next three final layers were a Dense, a Dropout and another Dense layer with softmax activation, forcing the output of the network to sum up to one. The dropout layer will randomly assign 0 weights to the neurons in the network. We have chosen a rate of 0.25 and thus 25% of the neurons will receive a weight of 0. In the model.fit function we chose a rate of 150 epochs. Initially we had set the value of 20 as the number of epochs. Nevertheless, this resulted in several errors, since the size of the batch (53) was bigger than iterations of complete passes. Therefore, we decided upon a value of 150 because that provided no errors and leads to the highest accuracy rate. After tweaking parameters in the Convolutional layers, such as the kernel size, the size of the feature detectors, and the amount of hidden dimensions, an accuracy between 0.73 and 0.91 was established. The last part of the script saves the model for easy use for deployment. The

Model	***Accuracy***
Gradient Boosting	0.63
Random Forest	0.76
1D CNN	0.82

Figure 14 | Accuracies of tests against the validation dataset

Model	***Accuracy***
Gradient Boosting	0.69
Random Forest	0.69
1D CNN	0.59

Figure 15 | Accuracies of tests against the testing dataset

final script can be found in the DWAAI Github folder under the name ‘1D-Convolutional Network.py’

A table of our classification report for the training and validation phase of each of the three models can be found in Figure 14 and 15.

Testing

Three slightly adjusted models were created to test the data. These had a function implemented that gave a prediction of the probability of class 0 (Brabants), based upon the data that was separated from the training and validation data in the beginning. Running the following models ‘random_forest_testing.py’, ‘gradient_boosting_testing.py’ and ‘ID-Convolutional Network_testing.py’ yielded the results that can be found in Figure 14.

Having a small dataset makes it more likely that the accuracy of the test data is often lower than the accuracy of training and validation dataset, since overfitting is more likely to happen. This was also visible in our models, especially in our 1D-CNN model wherein the accuracy significantly dropped. Next to this, overfitting was also visible when we plotted our value loss of the training data in our epochs, in the same graph as the value loss of the test data in our epochs. The value loss significantly increased when the number of epochs went up.

Deployment Model

Based upon the results given on the test data, we concluded that our 1D-CNN was not accurate enough yet for full deployment. Since both Random Forest and Gradient Boosting yielded the highest, but almost similar, accuracy for the test data we deployed both models again through the use of a new script that was written and included real time classification on the probability of sounding like a Brabander. All the 3 models were implemented in this code, as well as the pre-processing. This script can be found under the name ‘deployment_code.py’.

In our case, we connected the same audio recorder as the one used to record our training set samples. Then, ReactJS delivers the audio as a .wav file, which is sent through a post-request to a NodeJS backend server that stores the file locally. Then, the server calls a ‘depolyment_code.py’ with the path to the file as an argument. This script preprocesses the audio file by extracting 120 MFFC bands in real time and stores it in an array. Then the script will load the saved versions of our models and provide a ‘probability’ for each code. These three probabilities will be returned to NodeJS and ultimately one of these would be chosen to be shown on the screen. When deploying this script among 6 (3 from Brabant and 3 Non-Brabant) students of Industrial Design at the TU/e, we tested both the probabilities for Random Forest and Gradient Boosting. While logging these values and checking whether the participant was actually from Brabant, we discovered that Gradient Boosting had a significantly higher performance rate then Random Forest and thus decided upon Gradient Boosting to use as our final model, that was ready for deployment.

Suggested Improvements and Future Steps

Automating Model Training

During the process of evaluating a sample generated by an installation participant, it is saved to disk. This means that at static intervals, or perhaps even in real-time, it is possible to retrain the model on the go using this data. This should make a difference eventually, especially as the CNN model benefits from being trained on large amounts of data. This could be a part of the automated pipeline, by creating a retraining script that pulls new samples, and then trains the models. This script could then be triggered periodically by a cron job.

The difficulty in this task is that in order to be able to use the sample for training, it needs to be classified to fall in the Brabants or Non-Brabants categories. Eventually, this work needs to be done by a human, but given the meta-data we have on a participant, it should be feasible to automate, or at the least pre-populate most of the classification. These samples could then be used for training in a separate set, until their classification is confirmed by a human being. This allows for rolling increments in model accuracy, which means we should be able to make updates to the model that increase the accuracy relatively easily and quickly.

Online Data Gathering

Given that the user-facing parts of the installation are scripted in web-languages and frameworks such as ReactJS, JavaScript and NodeJS, it is exceedingly easy to deploy the installation online via a website or via native applications (using React Native). If done successfully, this would enable automated parallel data gathering from users throughout the Netherlands. Given how a convolutional network benefits more from large amounts of data than the other models, this would allow for better accuracy.

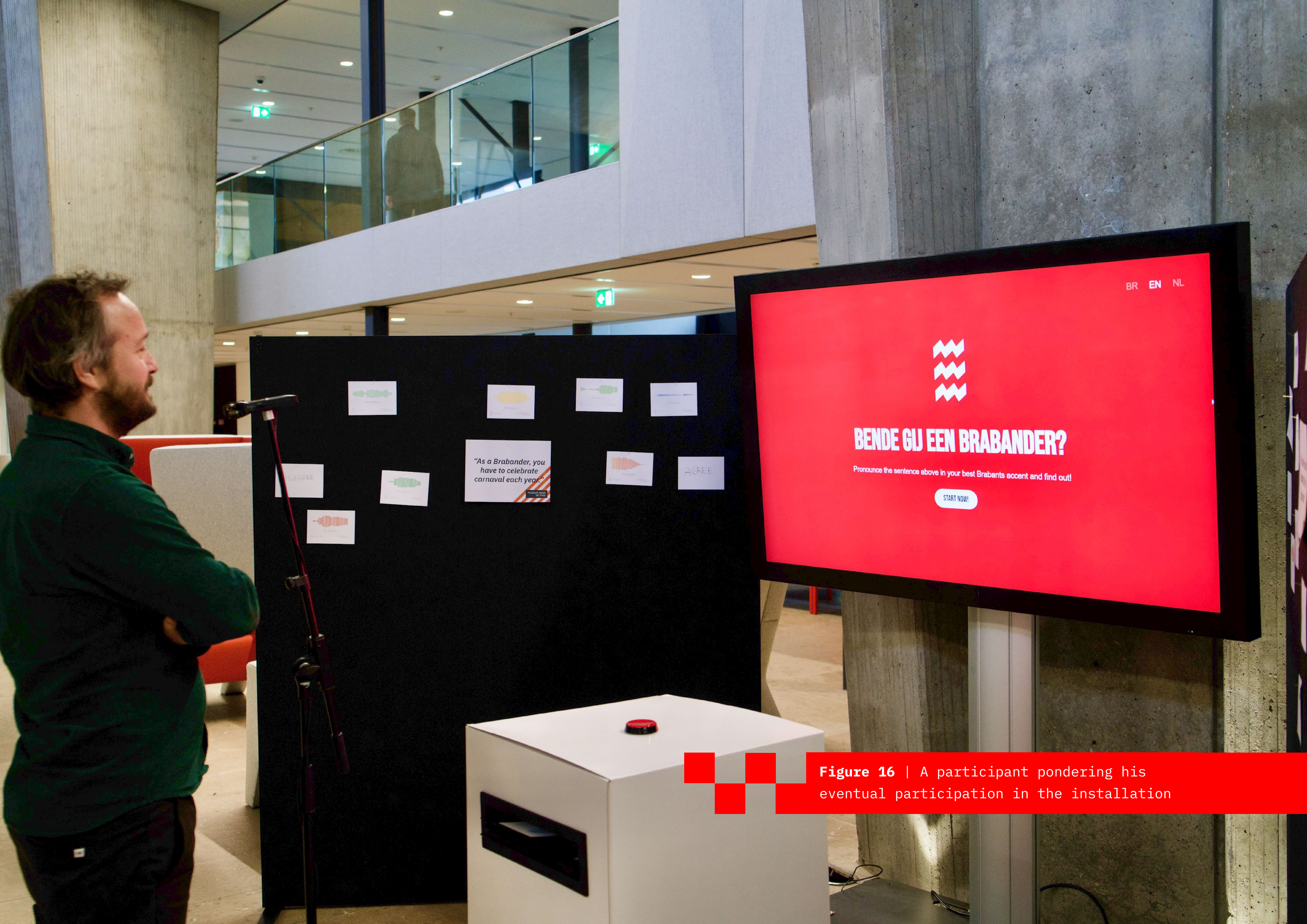

BR EN NL
BENDE GIJ EEN BRABANDER?
Pronounce the sentence above in your best Brabants accent and find out!
START NOW!
"As a Brabander, you have to celebrate carnaval each year."
DISAGREE
AGREE

Figure 16 | A participant pondering his eventual participation in the installation

Demonstrator

Features & Aesthetics

To give the installation this specific Brabant-vibe, everything is designed in red, white and black colors. This applies to both the design of the physical installation and the design of the interface. The interface has a very minimalistic look to make sure that visitors only look at what is important. A big headline shows the main title ("Bende gij een Brabander?") with a subtitle below that explains what to do ("Pronounce the sentence above in your best Brabants accent and find out!"). A button can be used to navigate to the next step (which in the physical installation is a big red button). The only extra functionality is a language switch at the top right, where users can use the installation in English, Dutch, or the local Brabant dialect. The translation to the Brabants dialect was validated by a native speaker.

As described earlier, before the visitor uses the installation, he or she has to scan a personal QR code. Then, when clicking the button for the first time, users can listen to a spoken example. A short karaoke-animation indicates that you as a visitor are expected to pronounce the sentence in a similar way. Then, users can try it themselves. After a countdown from three to zero, audio starts recording at the background for about 2.5 seconds. At the same time, a live visualization of the spoken audio by the visitor is visible. Visitors can then either retry or stop. A card is printed which visitors can then use to share their opinion on the statement that is visible at the board.

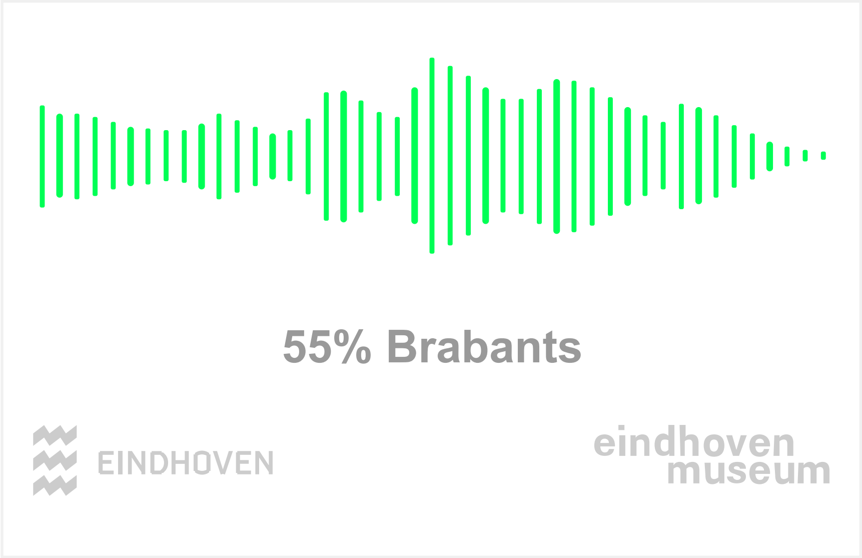
On the card, a couple of things are printed. First, it shows the visualization of the spoken audio. The color is generated according to the visitor's score to create a board with a colorful collection of cards. Also the score is printed on the card, together with two logos of both the City of Eindhoven and the Eindhoven Museum.

For a full overview of equipment and accessories used to build the envisioned installation as seen in Figure 16, see Appendix D.

Development

ReactJS was used for the development of the front-end which shows all visual parts of our installation. The application is responsive and can be used in every browser that supports playing and recording audio (possibly through a connected device). To be able to scan a participant's QR code, we used a webcam. For the audio recording, we called the default audio recording functionality available in all major browsers. Whenever an external audio recorder is connected, most browsers can be set-up in such a way that this input is used instead of the default computer recorder. In our case, we connected the same audio recorder as the one used to record our training set samples. Then, ReactJS delivers the audio as a .wav file, which is sent through a post-request to a NodeJS backend server that stores the file locally. As described in the 'Deployment Model' chapter, the server then calls a Python script with the path to the file as an argument. The Python script processes the audio and returns results, which is then shown in the front-end. Finally, the front-end generates the card with the final result and calls the printer to print the card.

A visual overview of the entire pipeline can be seen in figure 6 below. As described in the 'Deployment Model' chapter, instead of CNN we settled for Gradient Boosting for our tested prototype. However, CNN would be used in a future, ideal scenario.



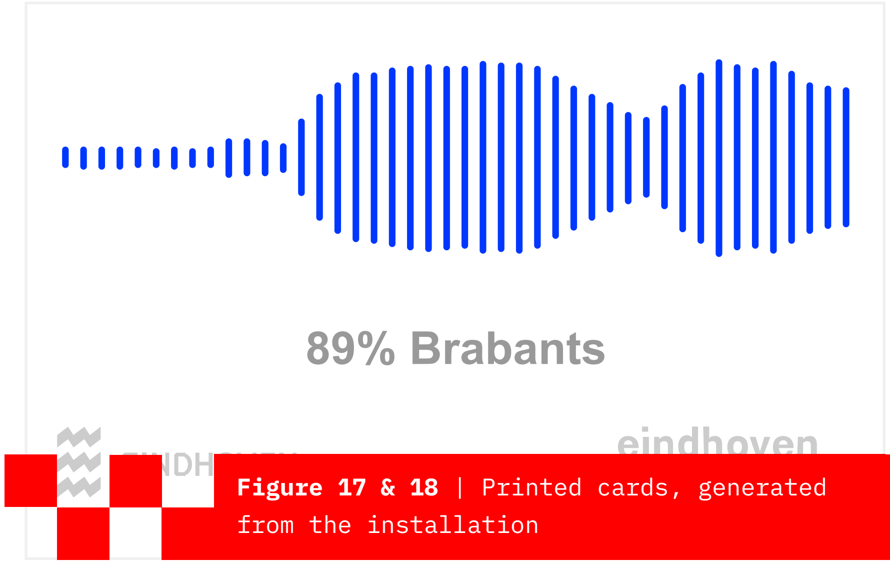


Figure 17 & 18 | Printed cards, generated from the installation

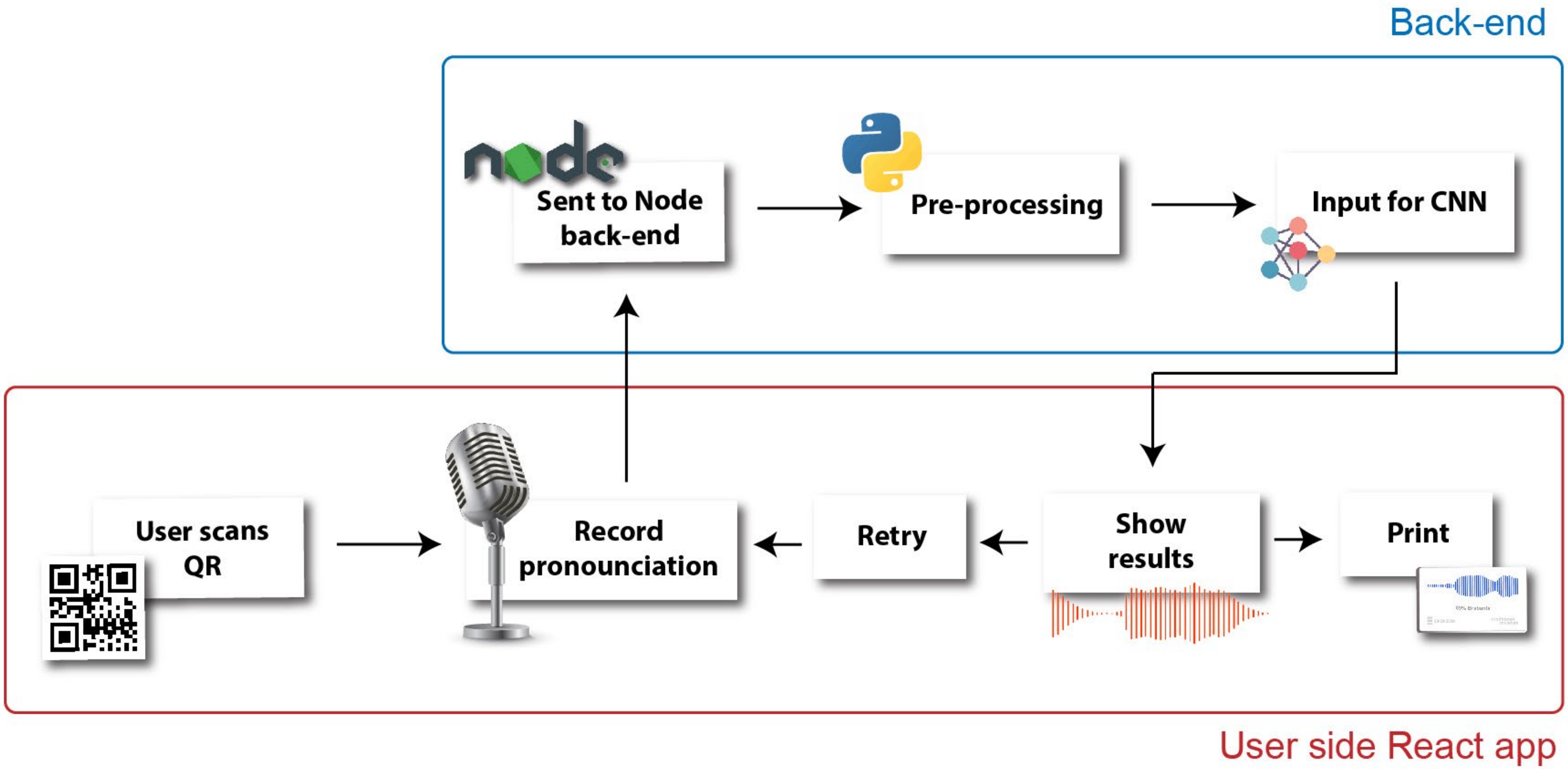


Figure 19 | An overview of the data flow in the installation application

Discussion

Connected Installations and Data

Given that the installation is situated in an environment with multiple other installations, which also produce data, it becomes feasible to look at how this data could be incorporated into our installation. Since our model is focused on detecting accent and not necessarily if someone is a Brabander or not, most data that other groups provide is not of direct use. However, most groups ask certain questions on residence, which for example could very well correlate with being a Brabander.

Therefore, a unique opportunity arises to benchmark the speech model with data about participants from other installations. By aggregating all features from all participants, we could determine which features correlate most with having a Brabant accent. And conversely, given a static definition of being a Brabander (for instance living in Brabant for five years), we can calculate how much this correlates with actually speaking like a Brabander. This affords particular quantitative clues about what this Brabantish identity could look like.

Facilitating Discussions

Reflecting on the original goal of the installation, let us look again at how the desired discussion is facilitated by the system. The initial desire was that interacting with the installation would spark conscious thought of what it means to be a Brabander. Although logistically difficult to achieve, the ideal installation would have a discussion guide present to help direct the process. Conceptually, if a single person interacts with the system, they would simply be engaged by the simplest form of the discussion, namely by placing their opinion on the presented statement. Depending on what the statement is at that moment in time, this in itself could spark a lot of reflective thought, but there is another decision to be made. In the presented prototype two different discussion topics were featured, but only a single card is printed per person. This provides an additional dynamic for users where they not only present their opinion on a topic, but also have to decide on what topic they prefer their voice to be heard. Both of these decisions become even more interesting when there are multiple visitors at once. The present guide can in this case start a discussion between the different participants on the topic they chose or by asking more general questions.

With the installation placed on the ground floor of Atlas, these theories were put to the test with some real volunteers. It was found that discussion was very much stimulated, but was mostly concerning the classification rather than the discussion topics. Mostly, discussions centered around what aspects of

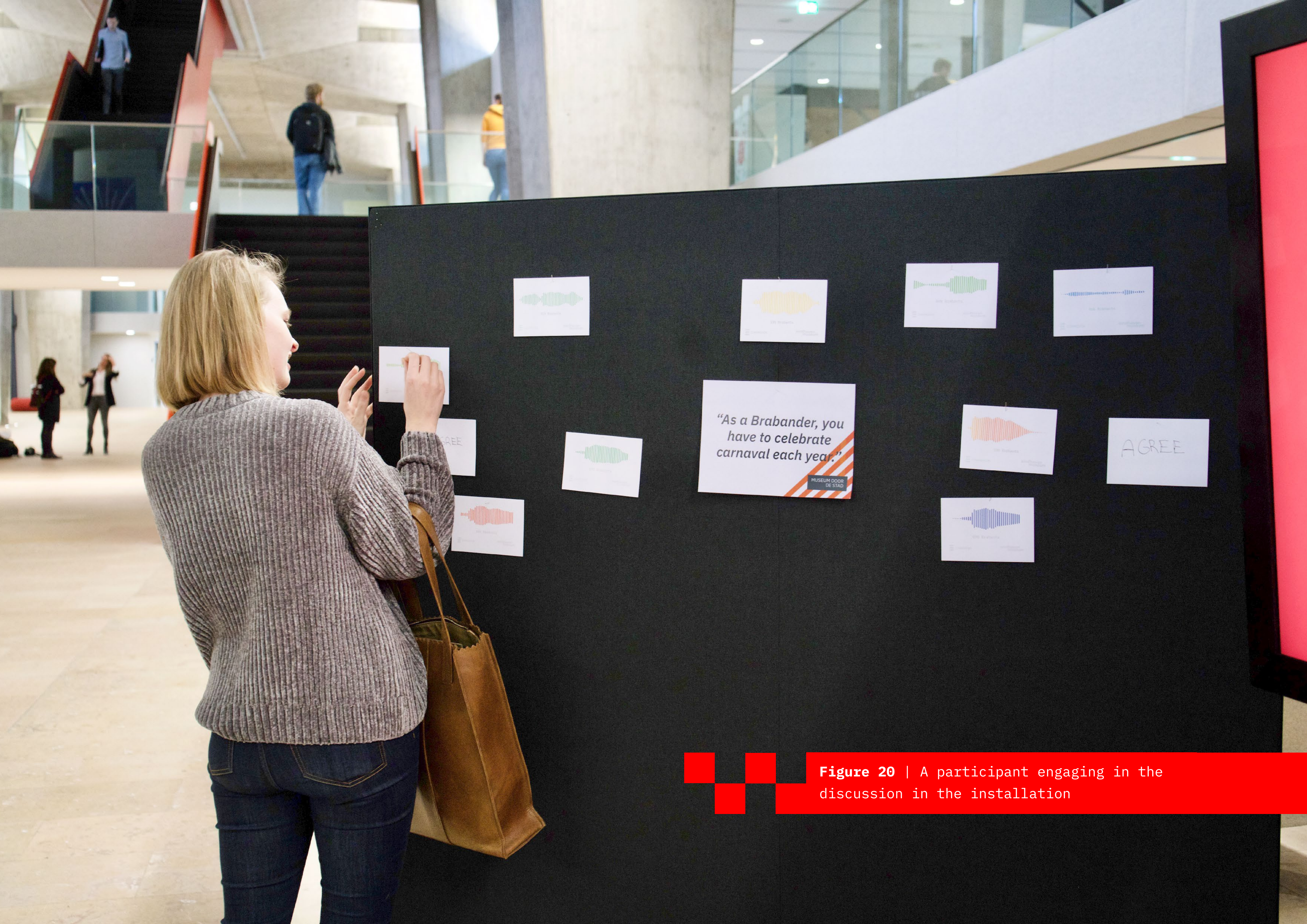


Figure 20 | A participant engaging in the discussion in the installation

the dialect had contributed to certain scores, which ended up being an enjoyable mystery for the visitors. At the time, two statements were pinned on the board: “As a Brabander you have to celebrate carnival each year” and “You have to speak Brabants to be a Brabander”. Out of these, the first one got nearly all attention with but a single exception, which in itself is an interesting finding that showcases what people enjoy discussing or what they find important.

Role of Designers in AI

Incorporating human-centered thinking to only some conceptual and visual aspects of the design process does not justify the importance we have as designers. Especially with the development of applications infused by Machine Learning, designers play an important role throughout the entire process. Human-centered thinking shall be used in every part of the design- and development, including data collection and the development of algorithms. This requires a deeper understanding of the AI field by designers. With that, it does not mean that the disciplines of data science and design should merge, but rather that both should work together and learn from each other to generate better outcomes [20].

Data in itself is static and raw. It is up to designers and developers to decide what kind of data is collected and how it is used and processed. Applying a user focus enables starting from a user needs perspective. This is where designers make a difference. We can look broader at data collection. Instead of just picking a certain type of data and sticking with it, we can think about outcomes that do serve user needs. We can question what data is retrieved and design transparent feedback to users. Moreover, we could think and validate how users respond to collecting their information. We serve as the voice of the user throughout the entire process.

To make building experiences work, designers require different mindsets and points of view. Designers partly build installations, they aim for high quality, they think about communication and framing, and, as mentioned, consider the collection of data. The big questions are: what do users want, what do we collect, and how do we inform users? In our concept, the installation aimed at sparking the discussion: what is important to be considered a Brabander? Through showing voice data that we used as the decisive factor in a transparent way, we discovered if users felt comfortable using voice to be classified as Brabants or not. As mentioned in the previous chapter, ‘Facilitating Discussion’, this resulted in interesting conversations. Using data & AI as creative material and as a means to trigger discussion, we tried to discover people’s opinions. These installations as a starting point for debate is another tool that designers could apply to their design processes.

Role of AI

Artificial Intelligence arguably plays a significant role in the eventual design of the installation. Reflectively looking back, the question on why AI is an indispensable part of the design comes to the front. The most interesting aspect here is related to the emergence of identity, both retrospective and futurospective, which is the core topic of the Eindhoven Museum, in particular in their latest iteration of Museum door de Stad.

We would argue that in these types of exhibitions, next to exploration of historical contexts, some critical analysis is required in order to place the historic context in a unique personal perspective, especially when the context being explored is as divergent and ever-changing as identity is. This was a particular aim in the initial stages of formulating the design.

Conversely, the concept of what a Brabander is, is of course highly dependent on personal values and biases. These factors can be static (appearance, origin, personality, etc.), or dynamic (social behaviour, attitude, values, etc.). Speech is special in this respect, because it is usually static, and depends on the region where one grows up. But with great effort, even more so for non-native speakers, it is possible to change one's speech to match a new region. But is this important, or even required? Should we require newcomers to adjust their way of speaking order to fin in?

In this regard, we consider the AI to be the mirror through which participants look at identity. If it is possible to categorise humans into belonging to a group or not, which factors are important, and which factors are not? More importantly, should group identity be based on any static definitions at all?

Finally, this approach also shows the human bias that is nearly always present in an AI. A machine cannot interpret speech to belong to different classes without a human guide interpreting the input. This realisation should reinforce the idea that this power of identity is distinctly human after all and help participants understand how nearly any AI is biased, prompting critical thought on the application of AI in societal contexts.

References

[1] Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition. 5.

[2] Nils Ackermann. 2018. Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences. Medium. Retrieved January 23, 2020 from https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf

[3] Jon Bunting. Optimizing an Accent Classification System. 7.

[4] Jeff Dayton-Johnson. 2003. The economic implications of social cohesion. University of Toronto Press.

[5] Andrea DeMarco and Stephen J Cox. 2013. Native Accent Classification via I-Vectors and Speaker Compensation Fusion. (2013), 5.

[6] Allen B Downey. 2016. Think DSP: digital signal processing in Python. O'Reilly Media, Inc.

[7] Yishan Jiao, Ming Tu, Visar Berisha, and Julie Liss. 2016. Accent Identification by Combining Deep Neural Networks and Recurrent Neural Networks Trained on Long and Short Term Features. 2388–2392. DOI:https://doi.org/10.21437/Interspeech.2016-1148

[8] Geena Kim. 2019. libphy/which_animal. Retrieved January 26, 2020 from https://github.com/libphy/which_animal

[9] Maryam Najafian, Saeid Safavi, Phil Weber, and Martin Russell. 2016. Identification of British English regional accents using fusion of i-vector and multi-accent phonotactic systems. 132–139. DOI:https://doi.org/10.21437/Odyssey.2016-19

[10] Lei Nelissen, Eva van der Born, Jordy Alblas, and Almar van der Stappen. 2020. leinelissen/dwaai. Retrieved January 23, 2020 from https://github.com/leinelissen/dwaai

[11] Marc van Oostendorp. 1997. Harde g en zachte g. Onze Taal. Retrieved January 23, 2020 from http://www.vanoostendorp.nl/fonologie/hardeg.html

[12] Carol Pedersen and Joachim Diederich. 2007. Accent Classification Using Support Vector Machines. In 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), IEEE, Melbourne, Australia, 444–449. DOI:https://doi.org/10.1109/ICIS.2007.47

[13] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. Scikit-learn: Machine Learning in Python. Mach. Learn. PYTHON, 6.

[14] Leon Mak An Sheng and Mok Wei Xiong Edmund. 2017. Deep Learning Approach to Accent Classification. (2017), 6.

[15] Harshdeep Singh. 2018. Understanding Gradient Boosting Machines. Medium. Retrieved January 24, 2020 from https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

[16] Dick Stanley. 2003. What Do We Know about Social Cohesion: The Research Perspective of the Federal Government's Social Cohesion Research Network. Can. J. Sociol. Cah. Can. Sociol. 28, 1 (2003), 5. DOI:https://doi.org/10.2307/3341872

[17] Dick Stanley and Sandra Smeltzer. 2003. Many happy returns: How social cohesion attracts investment. Econ. Implic. Soc. Cohes. (2003), 231–240.

[18] Jos Swanenberg and Jolijn Meulepas. 2011. Het nieuwe Brabants. Een onderzoek naar diversiteit in taal en cultuur onder jongeren in Noordoost-Brabant. Taal En Tongval Tijdschr. Voor Taalvariatie (2011), 303–328.

[19] Sandra Wagemakers. 2017. Brabant is Here: Making sense of regional identification. Tilburg University, Tilburg.

[20] Jon Wettersten and Dean Malmgren. 2018. What Happens When Data Scientists and Designers Work Together. Harvard Business Review. Retrieved January 26, 2020 from https://hbr.org/2018/03/what-happens-when-data-scientists-and-designers-work-together

[21] 2018. De mensen van Eindhoven. Centraal Bureau voor de Statistiek, The Hague, the Netherlands.

[22] 2018. Random Forests Classifiers in Python. DataCamp Community. Retrieved January 23, 2020 from https://www.datacamp.com/community/tutorials/random-forests-classifier-python

[23] 2019. Friezen, Zeeuwen, Drenten en Groningers meest trots op hun provincie. I&O Research. Retrieved January 21, 2020 from https://www.ioresearch.nl/actueel/1471-2/

[24] Over Eindhoven Museum. Eindhoven Museum. Retrieved January 21, 2020 from /nl/eindhoven-museum/over-eindhoven-museum

[25] Over Museum door de Stad. Eindhoven Museum. Retrieved November 26, 2019 from /nl/museum-door-de-stad/over-mdds

[26] Eindhoven Museum - Museum door Woensel. Eindhoven Museum. Retrieved January 21, 2020 from /nl/museum-door-de-stad/agenda/museum-door-woensel

[27] Nieuwe boomsoort in het Sprookjesbos: de Babbelboom. Retrieved January 24, 2020 from https://www.efteling.com/nl/blog/nieuws/20180924-babbelboom-sprookjesbos

[28] Taalschrift | Reportage | ABN was vooral een Hollandse uitvinding. Retrieved January 23, 2020 from http://taalschrift.org/reportage/000659.html

[29] machine learning - Is there a rule-of-thumb for how to divide a dataset into training and validation sets? Stack Overflow.

Retrieved January 24, 2020 from https://stackoverflow.com/questions/13610074/is-there-a-rule-of-thumb-for-how-to-divide-a-dataset-into-training-and-validatio

[30] Speech Recognition — Feature Extraction MFCC & PLP - Jonathan Hui - Medium. Retrieved January 24, 2020 from https://medium.com/@jonathan_hui/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9

[31] 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.22.1 documentation. Retrieved January 23, 2020 from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Appendices

Appendix A: GitHub Repository (includes code, data and models)

Appendix B: Design for the Museum tickets

Appendix C: Design for the theses

Appendix D: Overview of needed equipment for installation

- A wall to build a dedicated space for the installation
- Two big cork boards to put cards on
- A red carpet in front of the installation
- A microphone to record audio
- A speaker to play the audio file example
- A large screen to show the interface
- A printer to print the cards
- Set of empty white A6 cards
- A button to navigate between screens
- A statement and agree-to-disagree-spectrum printed and sticked to the cork boards
- A standard at arm height with the button on top of it and the printer hidden inside
- A small camera for the QR code scanner
- A stable internet connection to store results in the database (such as Data Foundry)
- Potentially a presenter dressed in Brabant clothing to engage visitors