2015•2016
FACULTEIT WETENSCHAPPEN
master in de informatica

Masterproef
Voice and gesture control system for TV

Promotor :
dr. Jan VAN DEN BERGH

Yunus Altin
Scriptie ingediend tot het behalen van de graad van master in de informatica

universiteit hasselt | Maastricht University

2015•2016
FACULTEIT WETENSCHAPPEN
master in de informatica

Masterproef
Voice and gesture control system for TV

Promotor :
dr. Jan VAN DEN BERGH

Yunus Altin
Scriptie ingediend tot het behalen van de graad van master in de informatica

Acknowledgements

First of all, I would like to thank my promoter dr. Jan Van den Bergh for all the time, feedback and advice he has given me along the way.

Also, I would like to thank Zappware for giving me the opportunity to be an intern at their firm. I would like to thank Koen Swings and Marc Vervoort for their advice and help during my internship. I also would like to thank Marc Van Daele for the technical support.

Thank you to the education team for all the knowledge during my years of study. And thank you, to everyone else, who made it possible for me to complete this education.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study, writing this thesis and my whole life. This accomplishment would not have been possible without them.

Abstract

Nowadays we are using a lot of different interaction techniques in our daily lives, like touch interaction, gestures and voice recognition but while everything evolved over the years interaction with a TV is still the same. Lately some television manufacturers are experimenting with new interaction techniques like gesture and voice but it's not clear if users enjoy using these new interaction techniques. In this master thesis we will implement two new interaction methods to the existing system of Zappware, voice-search and gesture recognition. This is implemented as an assignment of an internship for Zappware. We use an Android application for voice-search to recognize, analyse and send results to a set-top box and a ring device for gestures. To know the preferences/thoughts of users and to evaluate the system we have performed a user study with surveys before and after the experiment. The results shows that this system is better than users expected, and that they would use this system in their living room.

Contents

Chapter 1

Introduction

1.1 Motivation

Watching TV is a daily activity in our lives, while everything has evolved over the years: content type, content accessibility, and underlying TV technologies. Interaction with the TV didn't evolve that much, the most dominant interaction method with the TV is still the infrared remote control. This method is quite limited if we compare it to other interaction methods that we know of other devices like touch-based mobile devices or mouse-based PCs. Research on online video and interactive TV experiences is a growing field of study subdivided into topics that include: content production, system design and architecture, implementation, application and evaluation of viewing experiences, and novel interaction concepts [45]. Users expect better designed input devices, they expect information on the input device and expect other interaction methods like a trackball or speech recognition [27].

There are a lot of television manufacturers which provide new interaction techniques with the TV, like gesture based or voice recognition. Therefore research on people's preferences of interaction with TV is important.
Until now there are researches on different interaction techniques. Most of the researches focus on a specific action, like typing with the help of gestures [46, 35, 37], using gestures only for navigation actions [54, 34, 30], or using voice interaction for simple TV commands [39].

Speech technology promises to be a way to remove the translation between user's desires and a system's actions. Users can simply think of what they want, and say it. But it's not possible for a speech recognizer to fully understand the meaning of a text [51]. The use of speech technology is expanding from mobile to a living room environment where a TV is 3 meters away from the user [38].

1.2 Thesis objectives

In this thesis we will investigate new interaction techniques and implement voice and gesture interaction to an existing application to control a set-top box. More specifically voice interaction for voice-search and gestural interaction for navigation and quick access on the TV. These interaction techniques are not innovative but understanding people's preferences and thoughts are important to validate these interaction techniques. Most of this thesis is done as an assignment of an internship at Zappware (see Section 3.2), their goal is to do research into new interaction technologies and offer a more natural way of interaction. A common case for voice interaction is inputting text for a search request which is currently done through an on-screen keyboard or digit-keys on the remote control. Simply saying a search request in natural language would greatly enhance the search functionality. The objectives are summarized as follow:

1. Voice and gesture interaction techniques
    - Voice-search and voice commands with natural language understanding;
    - 2-D gesture-based interaction for navigation and quick access to UI elements;
2. Mapping the commands with actions on a set-top box
3. Evaluate this system and understanding people's preferences and thoughts for interacting with TV

1.3 Structure of this thesis

In chapter 2 the evolution of the remote control is described, the history is summarized and the products of now both industrial and academic. In chapter 3 the implementation as part of the internship is described, you can also find an overview of the architecture and used devices. In chapter 5 the results of the user tests can be found. In chapter 6 the conclusion and a discussion about my findings and future work are presented.

Chapter 2

Evolution of the Remote Control

2.1 History

In 1893 the remote controller was described by Nikola Tesla in U.S. Patent 613809 [52], it was used to remotely control the motorboats during WW1 [26]. The early developments of the remote control started in 1901 by the Spanish engineer, Leonardo Torres Quevedo. He created a system in 1903 to do mechanical movements at a distance, named Telekine. This system was used as a way of testing dirigible balloons without risking human lives. This system is laid down the modern wireless remote-control operation principles. But the first time a remote control entered into our living rooms, it was the "Lazy Bone" of The Zenith Radio Corporation [26]. It could only change channels and turn the television on/off. It was not even wireless, it was attached to the television with a cable. But with the advertisements (see Figure 2.1) they gave the users a feeling of relaxation and laziness.
After the Lazy Bone, the remote controllers were not seen as a device of relaxation but more as a way to fight back against excesses of advertising. The users wanted to protect themselves against brainwashing advertisements [32]. First came the TV Hush and Blab-off, it was a volume and a mute button that could be used from your seat.

In 1955, Zenith engineer Eugene Polley introduced a wireless remote control called "Flashmatic" [26], it was shaped like a gun and the user could mute the commercials (see Figure 2.11). This was the first industrial wireless TV remote, it operated by means of four photo cells (in each corner a photo

Now! Zenith Adds "LAZY BONES" Remote Control
to Black Magic Television
All New
ZENITH

Figure 2.1: Lazy bone advertisement [32]

cell). The device was like a flashlight that should be aimed at a sensor at the corner of the TV. They used different corners to control different aspects, one corner for the volume and the other corner to adjust the tuning. It was not a complex device, really simple to use but not so useful. For instance if there was sunshine on the TV, the tuner might start automatically rotating and switch channels.

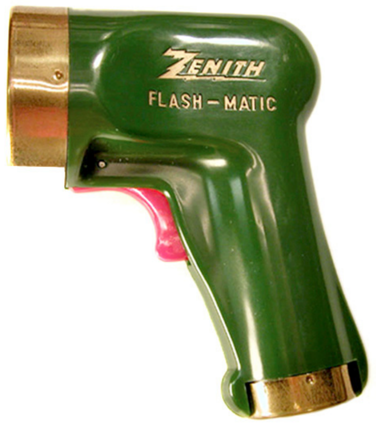


Figure 2.2: The Flash Matic remote control [1]

There were some development challenges before the ultrasonic remote controls. First they thought about radio waves, but this method would permeate the signal through walls and this would be a problem if the user lives in an adjacent apartment or room [20].
Zenith engineers also discussed the opportunity to use sound signals, but they believed the sound would disturb the users. The sales people of Zenith were against using batteries in the remote control, because they thought when the remote control wouldn't work anymore the users would think it's because it's broken instead of that the batteries are empty.

In 1956 the Zenith Space Command remote [26] (see Figure 2.3) was produced, designed by Robert Adler and was based on ultrasonic sound signals (high-frequency sound, out of the range of human hearing). The remote control didn't need batteries, it was built around aluminium rods that were very light, and they emitted distinctive high-frequency sounds. It had four rods, two for switch the channel up/down, one to mute the volume mute and one to turn the TV on/off.

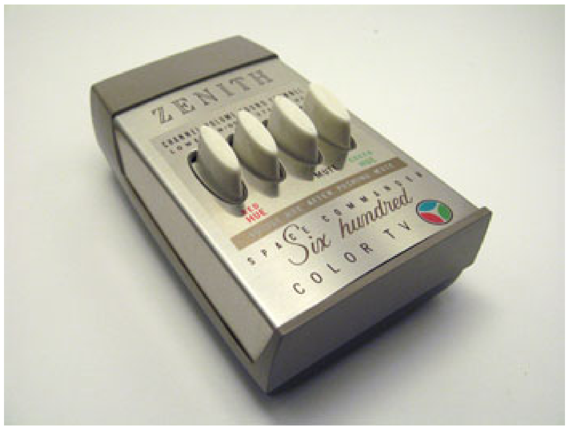


Figure 2.3: The Zenith Space Command remote [2]

This ultrasonic remote control was expensive because it needed an elaborate receiver (six vacuum tubes) to pick up signals. In 1960 the vacuum tubes were replaced by solid-state circuitry. It was now possible to create battery-powered remote control. The ultrasonic remote controls lasted until 1980.

In the early 1980s the ultrasonic remote controls were replaced by the infrared remote controls, which we use now. IR remote uses a low frequency light beam, which can't be seen by the human eye but can be detected by the receiver in the TV [29]. In 1985 Philips introduced the first universal remote (U.S. Patent #4774511 [47]). This remote control allowed to control different types of appliances from different manufacturers. The user selects the type of appliances, manufacturers and the model number. The selection elements are scanned automatically with a microprocessor and there will be a correspondent address generated. With this address it's possible to get the parameters of the product: pulse width, frequency, pulse repetition rate, word length, etc. and it can transmit the corresponding signal to the controlled appliance to execute an action selected by the user.

The first wireless remote control with a trackball [4] (see Figure 2.4) is invented by Zenith in 2001 (patent #USD381661). This remote control worked like a mouse, if the user clicks on the ball a cursor was shown on the screen. If the user rolls the ball, hidden menus in corners will be activated and shown. Then the user can activate options from this menu, etc. In 2011 it was possible to use your smartphone for example an iPhone to control a TV via Bluetooth.

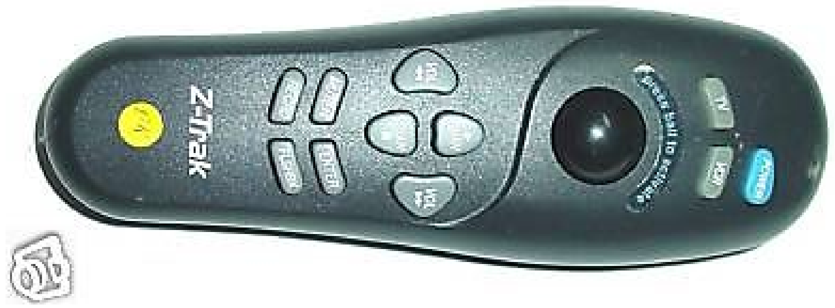


Figure 2.4: The Z-Trak [3]

Samsung [8] introduced in 2012 a remote control with a touchpad and a voice recognition function. The Samsung Remote control TM1290 had only 12 buttons. They updated this model one year later with a touchpad where users could scroll in four directions and input number with a finger. There were also more shortcuts for smart functions, this brought the number of buttons to 23. One year later in 2014, they changed the whole design. It now has a rectangular shape to an elongated pebble-like oval design that fits more naturally in the user's hand. The most important features are the motion gestures (to quickly access some elements of the UI), the touchpad and the voice recognition.

LG introduced the LG Magic Remote (see Figure 2.5) in 2013 [5], this remote control offers four modes of interaction. The user can interact using voice, gestures, wheel, and by pointing the remote control. The user can switch channels with the wheel, and use the voice recognition to browse the web for example. LG still uses this remote control nowadays with little improvements.

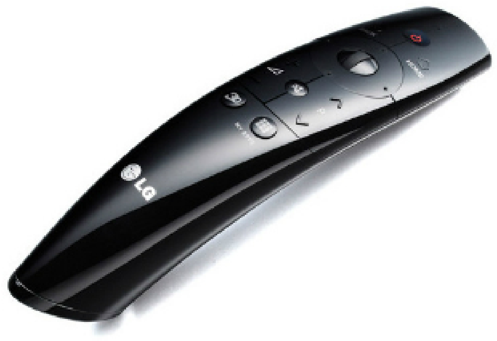


Figure 2.5: The LG Magic Remote [5]

2.2 Nowadays

2.2.1 Apple TV

Apple TV [15] has a button on his remote control for voice-search (see Figure 2.6), if the user pushes the button Siri [23] will be activated. The user can simply say what movie or show he/she wants to watch. With the help of dual microphones, the remote control can hear the voice clearly, even when there is background noise. The user can search for a movie/TV show across popular services, like Netflix and iTunes. By pressing and releasing the button Siri [23] will display suggestions on the screen. The user can search movies and TV shows by title, genre, cast, and more (e.g. "I want to watch Gone Girl", "Show me some funny horror movies", "What are some popular new releases?"). It's also possible to refine the search results by actor, time period, director, and more (e.g. "Only the good ones", "Only comedies"). The user can further search in Apple Music, App Store, check the weather and more. The user also can do some voice commands with Siri like play/pause, rewind/fast forward, skip back 15 seconds and temporarily turn on captioning when asked "What did he say?" or "What did she say?", "Open a specific app", and more. The remote control also has a glass touch surface for navigation around the GUI, the user can swipe left/right/up/down, tap on the screen to select or scrub to fast forward or rewind.

2.2.2 Android TV

The voice-search of Android TV [14] starts when the user presses the voice key on the remote control. The user can then do some actions like change the channel to a specific channel for example BBC or go directly to a TV show/movie, search by description, play YouTube videos, launch an app (for example "Open Netflix"), go to a website, do a web search, etc.

The usage of the voice-search is as follow, the user presses the voice key on the remote control or on the Android application. This will wake up the TV, the microphone button will become red (see Figure 2.7). The speech text will be shown with a continuous feedback design, where every word that's spoken will be shown immediately.
Some example actions and queries [13]:

- Change the channel (e.g. "NBC", "BBC news")
- Go directly to a TV show/movie (e.g. "Modern Family", "Cars 2")

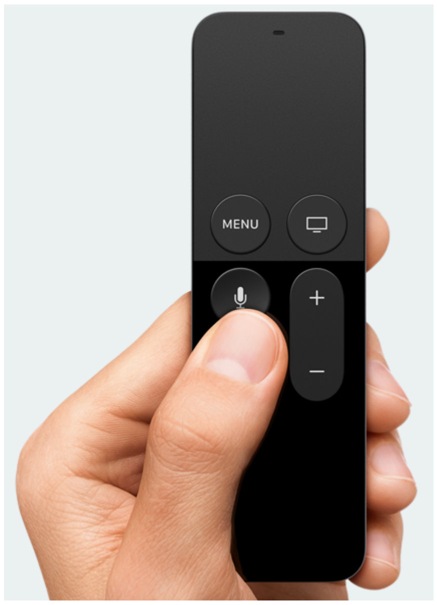


Figure 2.6: The Apple TV remote control [15]

Figure 2.7: Android TV [56]

- Search by description (e.g. “TV show about football”, “Movies with Jeff Bridges”)
- Play YouTube videos (e.g. “Lady Gaga music video”, “How to throw a spiral”)
- Launch an application or shortcut key (e.g. “Open photos”, “Open Netflix”)
- Go to a website (e.g. “Twitter.com”, “Amazon.com”)
- Do a web search (e.g. “Google weather”, “Google restaurants”)

The user can also force the Android TV to recognize the right commands by adding prefixes:

- “Channel...” switches to that channel on Live TV
- “Watch...” finds a specific channel, TV show or movie title
- “Search...” uses Google TV Search
- “YouTube...” searches for or plays a YouTube video
- “Google...” does a web search
- “Open...” opens an application or web page

Sony Bravia’s Android TV of 2015 [22] has a remote control with a voice-search button and a touchpad for simple swipes like swiping left, right, up and down to navigate in the menu. You can also use the touchpad as a mouse while using an internet browser on the TV.

2.2.3 Amazon Fire TV

The Amazon Fire TV Remote [7] is simple and has all the controls the users need to search, navigate, watch and play games (see Figure 2.8). Bluetooth is used to connect the remote control with the TV. The most interesting button on the remote control is the voice button. With this button it’s possible to do voice-search, the user can search movies, TV shows, actors and genres using their voice. The voice-search is supported for the Amazon video, app, game catalog, and for Hulu plus, HBO GO, Vevo and Showtime Anytime. But it doesn’t work for individual apps, for example for Netflix.

The voice-search [11] only starts when the user pushes the button, thus it's not possible to trigger the voice-search with a sentence like "Hi TV". The user has to hold the button also when he/she is still speaking and release the button when he/she is done. The Amazon Fire TV doesn't support natural language commands, the user can only say a movie title, TV title, actor name, character name, director name, or genre instead of saying the whole sentence like "Show me all comedy movies". The user can then use the directional pad to see the search results and he/she can select an item by pressing the "Select" button.

Figure 2.8: Amazon Fire TV: Voice-Search [9]

2.2.4 Samsung Smart TV

Voice Control

The Voice Control feature [12, 10] can be used with the remote control, a list of basic voice navigation commands supported by Samsung Smart TV are the following:

- Focus up/down/left/right: move the Focus to the object above/below/left/right
- Select: select the focused object
- Exit: exit application

The user can control the TV with commands like "Volume up", "Channel up", "Volume <number>",... The user can access some apps by saying the name of the services, e.g. "Smart Hub" (see Figure 2.9).
The Voice Interaction feature enables the natural language understanding of the TV, this is only supported in a few locations but Voice Control with pre-set voice commands is supported in all locations.
The Voice Mode will be triggered when the user presses the Voice Mode button on the Samsung Remote control or by saying the trigger sentence which is by default "Hi TV". Depending on the settings and location, Voice Interaction or Voice Control will be activated.

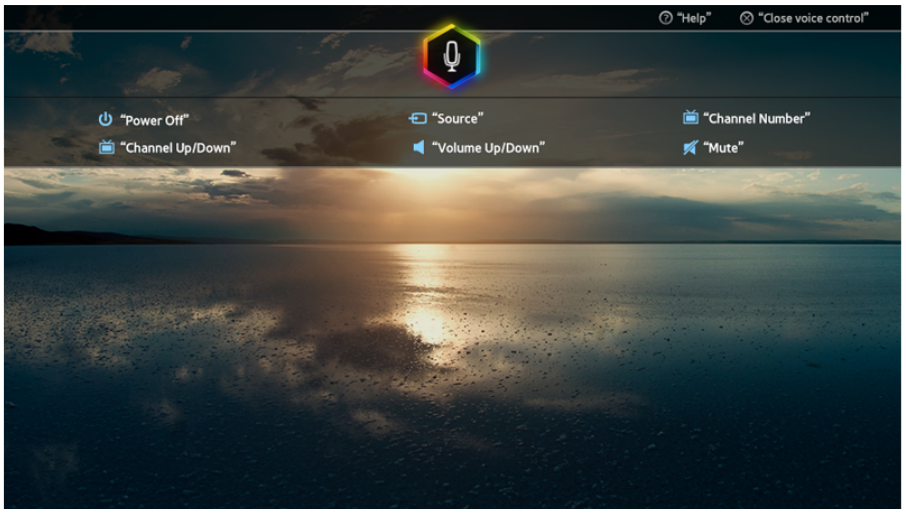


Figure 2.9: Samsung Smart TV Voice Control [12]

Gesture-control

In the Samsung Gesture book, there are 13 gestures possible to interact with the TV. The first one is waving your hand, this is the initial gesture that indicates the user is going to use gestures. It's similar to the voice mode were they use "Hi TV" to trigger the voice recognition. You can flip to the right or the left to go to a next or previous page in the menu, flip to the next/previous photo on the gallery or skip/backward 10 seconds in a movie. The user can move his hand freely and change the focus of the content, it's like a mouse pointer. With the grab gesture the user can "select" or enter content on the menu. With a long grab the user can change channels and control the volume. The user can perform the gesture grab and move to scroll in webpages, pictures, maps, etc. With the counter clockwise rotation gesture the user can undo his action, and just go back in the menu. It is also possible to us two hand by using gestures, first the user has to wave with two hands this time to indicate for using two hands gestures. To zoom in/out the user has to use both hands for the gesture "grab and widening/narrowing". To rotate photos the user can grab and rotate with both hands. The user can do a thumb up to automatically like Facebook content.

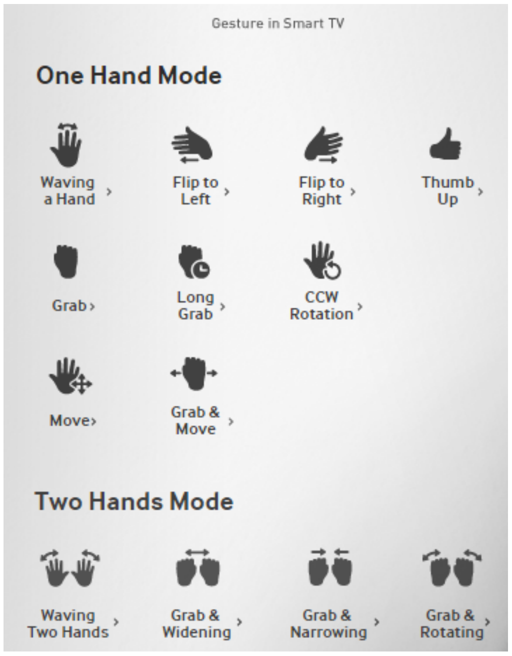


Figure 2.10: Samsung Smart TV all gestures [6]

2.3 Academic

2.3.1 PalmRC

PalmRC [31], imaginary palm-based remote control for eyes-free television is an academic research remote control by Technische Universitat Darmstadt (Germany). This remote control focuses on eyes-free, where the user doesn't need to switch their attention between the television and the remote control. It allows users to operate the TV with empty hands, so that they can keep their focus on the TV screen. They can interact by using swipes and pointing on their dominant hand. The system receives the position of the touches on the palm and gives feedback on the screen. The user can do the following commands: navigation on the EPG, volume up/down and direct interaction with the menu.

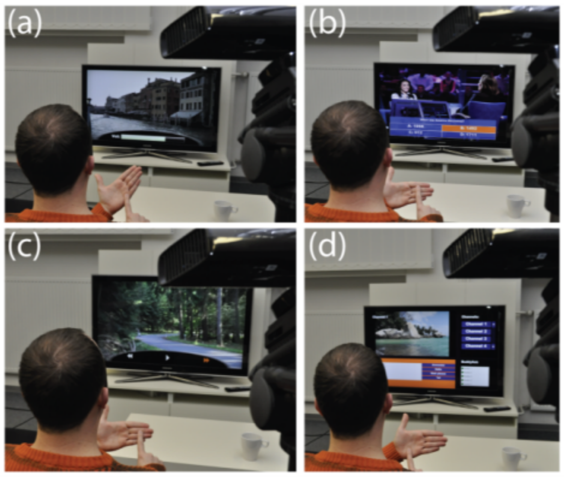


Figure 2.11: The PalmRC [31]

2.3.2 Leap Gestures for TV

Leap Gestures for TV [55], is an elicitation study of University Stefan cel Mare of Suceava. It's a presentation of insights from a gesture elicitation study in the context of interacting with TV. It's possible to do 21 TV commands (open, close, next/previous channel, volume up/down/mute, open/hide menu, help, yes/no, go to fav./2nd fav. channel, go to random channel, go to channel #7 or #27, TV Guide, show channels list, open

browser) with free-hand gestures in this study. Some industrial examples of gesture interfaces are the LG Magic Remote (see Section 2.1) and the Samsung gesture book (see Section 2.2.4). The focus of this paper is the preferences of the users for interacting with TV with free-hand gesture (3-D finger movements). They have asked 18 participants to find gestures by their self for the 21 TV commands and to rate them with a 5-point Likert scale for how well it fit, how easily they were able to remember and if they preferred the leap gesture or a TV remote button.
The participants used in average 20.5 seconds to find a suitable gesture command. The overall median of the gesture goodness was 4/5. The best rated gestures were "next/previous channel" and "volume up/down", while the lowest rated gestures were "go to fav./2nd fav. channel", "volume mute", "open browser", "and TV guide". The participants voted 82% that they would prefer gestures above TV remote buttons, 12% remote buttons and 12% neutral. The recall likeliness of the gestures was 72.8% while in 11.4% the participants couldn't remember the gestures.
The important findings of this experiment are:

- Finger and hand pose gestures are preferred to remote buttons, but there is a low agreement between users. This means that finger gestures should be personalized with a user-dependent training.
- Users fall back on previously acquired gesture interaction models. For example most of the users now use touch-screen devices and are familiar with the swipe gestures.
- Users prefer 2-D gestures.
- Users prefer either motion or hand pose gestures, and combinations of these two are less likely.
- Users associate gestures and commands to maximize the recall rate.
- Preference for culture-specific leap gestures, for example thumbs-up or hand wave.
- Exploit hand pose to distinguish commands.
- Users prefer to draw the first letter of task names in mid-air to execute it.
- Concrete or imaginary surface to assist users to articulate gestures.

We'll now describe only the best fitting gestures with the command, from the perspective of the participants and the authors. To open the TV they have chosen for an "open palm", to close a TV a "close palm". To go to the next channel, they have chosen to move the hand right to left or left to right, the same for previous channel but the opposite direction. To increase the volume they have chosen to move their palm upward and to decrease the volume to move it downward. The users can mute the volume by pinching their fingers together. To open the menu the participants have chosen to draw the letter "M" and to hide it they have chosen for a hand wave. To activate the help function, they have chosen the letter "H" or the symbol "?". To accept something, like a "yes" they have chosen a thumbs-up hand pose and for the "no" a hand wave. The gesture to go to the favourite channel is performed by showing the index finger, for the 2nd favourite channel the users have to show two fingers (index and middle). To go to channel 7 or 27 the users have chosen to draw the correspondent number in mid-air. For the TV guide they have chosen to draw the letter "G" and for the channels list a square. To go to the last channel they have chosen to draw a circle anti-clockwise. And the last gesture to open a browser the participants have chosen to draw the symbol "@".

2.3.3 Voice as sound: using non-verbal voice input for interactive control

Igarashi et al. [39] proposes an indirect technique of speech recognition. They used the speech recognition engine to turn speech into words or sentences, and the system performs actions based on the text. So the user has to complete the sentence/word and wait for the action. This technique is not direct, so they propose to use non-verbal features in speech like volume pitch and continuation.

Control by continuous voice: you have to think about a button being pressed when the user is making a sound, and the button will be released when he stops the sound. An example could be when the user wants to increase the volume he could say "Volume up, aaaaaaaaahh" and the volume will increase while the user say aaah and stops when he stops making the sound. The user can continuously observe the immediate feedback during the interaction.

Control by pitch: they also can use the pitch as a speed parameter, for example when we take the same example as above but with a pitch. When the user increases the pitch of his voice the volume of the television can increase faster. This could be handy when the user wants to scroll, and just by increasing the pitch he can scroll faster or vice versa.

Control by tonguing: the user can control the TV by using tonguing, for example when the user wants to increase the number of the channel (to zap 3 channels further). He just can say “Channel up, ta ta ta” and it will increase by three. The user can also use bodily actions like hand clapping and finger snapping to make noise instead of saying “ta ta ta”.
The advantages of this technique are immediate, continuous control, language independence and simplicity. But the limitation is that we have to use an unnatural way of our voice, it can also tire the throat after a while.

2.3.4 Designing natural speech interactions for the living room

Stifelman et al . [51] addresses three challenges in their research:

- How can we convey the scope of what the system can understand?
- How should we provide feedback of what the system understood without distracting the visual content?
- The role of text-to-speech on the living room, how should it be integrated with visual feedback?

Convey the scope by showing example sentences on the screen which the user can fill and change to his desire. In this user study they show that most of the users (44%) extrapolated from the examples with their own search criteria, 35% of them use keywords (phrases without verbs) such as “action movies”. Only 16% use a unique phrase and 5% use exactly the same as the example.

There are two types of feedback, one is called continuous speech feedback, and this occurs when the recognized words are displayed in real-time as the user speaks the phrase. The other type of feedback is called after-speech feedback, this occurs when the feedback is at the end of the speech. The participants of the experiment preferred the responsiveness of the continuous design better because it’s giving them immediate feedback, and the ability to detect errors rapidly and to see the feedback peripherally while still viewing the content. But they also found it confusing when the errors were corrected later in the text.

Spoken feedback is not preferred while interacting with a TV program guide when live TV is playing. They used a grey bubble that focuses on the current turn (TTS). In the example the user asks for movies with Mia Farrow and Woody Allen, this is shown on the screen (see Figure 2.12). Next the user asks “which ones are comedies?”, the system responds “here’re the ones that are comedies”.

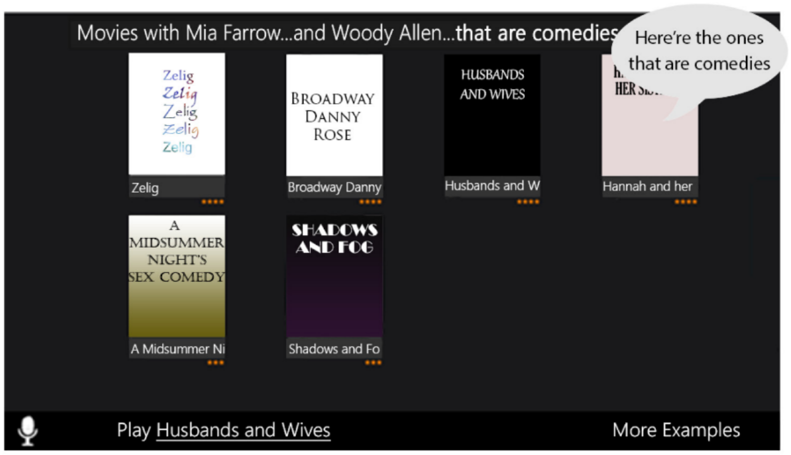


Figure 2.12: Visual feedback [51]

2.3.5 Prospects for Unrestricted Speech Input for TV Content Search

The remote control of Wittenburg et. al [57] has four navigation buttons, one button to trigger speech recognition, and a button for selection. The tasks of the participants are, for example:

- “Someone told you about a program about container ships on The Discovery Channel sometime this week. Can you find it so you can either watch or record it (depending on the schedule)?” In this case, “container ships” was in the title of the program, but not in the description.
- “See if there are any programs this week about cooking turkeys for

Thanksgiving." This task was very open ended, although there were a number of relevant programs.

- "Remember the episode of M*A*S*H where everyone's afraid that Captain Pierce was killed at the front? See if it's on this week." This task was presented either orally or on paper. It required participants to find the program M*A*S*H and then find an episode that was similar to the description, but used different words; this simulated a friend's recommendation or a dim recollection.
- "The FX program, Cops, features police officers in different cities. You used to live in Seattle–do they ever show that city's cops?" The word "Seattle" was in each of two episode names in the result set, but only in the description of one of them.

When the system recognized what the participants wanted to say, they enjoyed using the system. This is because successful interactions were very quick. If the program was first listed, they just push the select button without checking the other options.
They were comfortable with using the remote control as a microphone. But after pressing the speech recognition button they paused a while and were thinking about what to say, so it's important to automatically detect the onset of speech and the end of speech.

When the recognition failed, the participants tried some recovery strategies that failed in some way, some examples are:

- Repeating the same utterance, with slight changes in inflection.
- Using more or less of the title. Example: "Johnny Cash", then "I walk the line", then "Johnny Cash I walk the line".
- Changes in pronunciation
- Adding details, example: when "Friends" didn't work, one participant tried "Friends baby shower", adding detail from the episode title or description.
- Saying individual words instead of continuous speech
- Speaking more slowly
- Microphone closer to the mouth

Users are not interested in searching through a list of program names but were happier to look through episodes.

2.3.6 How Do Users Respond to Voice Input Errors? Lexical and Phonetic Query Reformulation in Voice-Search

Jiang et al. [42] did a research on how users respond to voice input errors, first we will explain the terms and summarize the reformulation patterns. There are two kinds of queries:

- **Voice query (qv):** what the user speaks
- **Transcribed query (qtr):** what the system recognized

There are two kind of recognition errors:

- **Missing words:** words in qv that don't appear in qtr
- **Incorrect words:** words in qtr that don't appear in qv

Voice input errors can be categorized into four categories:

- **Speech recognition error:** the participant completed a query without any interruption, but the voice query was not recognized correctly. This error can be characterized by missing words or/and incorrect words as mentioned earlier.
- **System Interruption:** the participant was improperly interrupted by the system and failed to speak all of the query words.
- **No Error:** no voice input error.
- **Query suggestion:** the participant used a Google's query suggestion. If the search history recorded that the participant searched for a query while we did not hear it in the recording, we consider that to be a case of using Google's query suggestion.

There are four lexical and two phonetic voice query reformulation patterns, summarized as follow:

Lexical Query Reformulation Patterns

- **Addition (ADD):** adding new words to the query.
- **Substitution (SUB):** replacing words with semantically-related words.
- **Removal (RMV):** removing words from the query.
- **Re-ordering (ORD):** changing the order of the words in a query.

Phonetic Query Reformulation Patterns

Partial Emphasis (PE): Partial emphasis refers to the behaviour of phonetically emphasizing a part of the current query that also appeared in the previous query.

STR	rap and crime	put stress on "rap"
SLW	rap and c-r-i-m-e	slow down at "crime"
SPL	Puerto Rico	spell out each letter in "Puerto"
DIF	P u e r to Rico	pronounce "Puerto" differently

Whole Emphasis (WE): Whole emphasis is to place emphasis on every part of the query, usually by putting stress or slow down on each of the words. It usually happens when the majority of the previous query was wrongly recognized.

	Voice Query	**Transcribed Query**
q1	art embezzlement	are in dublin
q2	a-r-t e-m-b-e-z-z-l-e-m-e-n-t	art embezzlement

Speech recognition errors affect the voice queries the most, it occurred in 810 voice queries (89.2% of all 908 queries). This will hurt the performance of voice-search. There were 1.77 missing words and 1.84 incorrect words per query, this also affects the voice-search results. Voice input error is an important issue that has to be resolved for voice-search. Their solution is to support users' query reformulation, this can be done by designing the interface so that it supports voice query reformulation and developing query suggestion algorithms.

2.3.7 SpeeG2: A Speech- and Gesture-based Interface for Efficient Controller-free Text Entry

SpeeG2 [37] uses a continuous feedback design, when the users speaks the words are visualized. After a sentence is spoken, the selection process starts. The user can then start to correct the recognized word sequence by using his/her dominant hand. These hand movements are registered by a Microsoft Kinect and are transformed to screen coordinates. The interface of SpeeG2 is shown on figure 2.13, every part of the UI has a number which are described below.

1. Direct feedback what the user says, which means that the words can

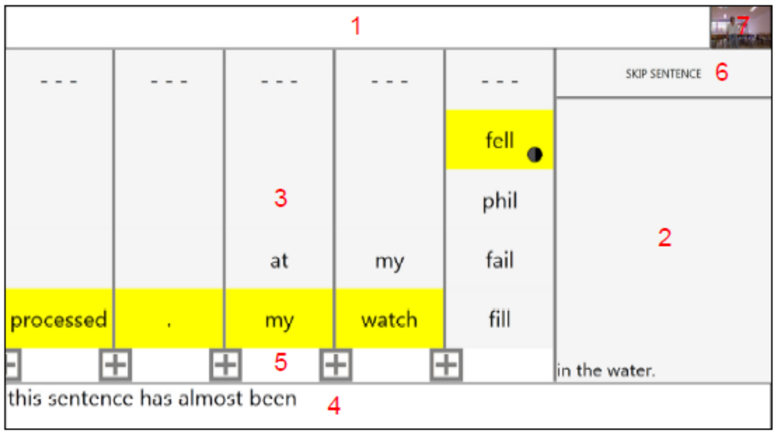


Figure 2.13: SpeeG2 interface [37]

change depending on the grammar rules. But the user get every word on the screen when he is speaking.

2. Here we see what still needs to be processed, in this example it's "in the water". This actually is like a queue of words that needs to be processed.

3. This is the grid where the processing is done, the row is the word sequence and the columns represent the alternative word candidates. The alternative word list is inverse in the column, the most matched word is the word on the bottom of the list. The user can confirm a correctly recognized sentence really easy with minimal effort. The user can either select another word or by selecting "- - -" he can skip and delete a particular word.

4. Here the user can see what is already been processed.

5. The user can also insert new words between two words/columns.

6. It's also possible to skip a sentence if the results are for example mostly incorrect. Only the current sentence will be deleted, this is the sentence with the most number of words in the grid. In this example it's the sentence "my watch fell in the water".

7. Camera visualization, to show the camera feed in order to enable users to correct improper positioning.

Prototypes:

1. **Scroller:** the words are scrolled at the grid, the speed of this scrolling is controlled by the dominant hand (horizontal movement). The user is also allowed to go back to the previously confirmed words by moving the dominant hand on the opposite direction. (13 WPM)
2. **Typewriter:** the selection of the words by typewriter prototype is done by a single swipe. (21 WPM)

They have also tested the use of speech only, they got a speed of 77.63 words per minute (WPM). With the current virtual keyboard solution we only achieve an average text input of 5.79 WPM, while the original keyboard gets 38 WPM.

2.3.8 Active Haptic Feedback for Touch Enabled TV Remote

Treskunov et al. [53] investigated how haptic feedback affects the user experience of the touchpad-based TV remote. Because due to the latency in visual feedback, there is a mismatch between the finger movement and the visual perception in the TV UI, this can cause overshooting. They conducted two user studies with two prototypes to evaluate the user preference and the user performance. For the first user study a QT based Nokia N9 and Android based Samsung Galaxy S3 were used (haptic actuators embedded). The participants (8 in total) used the smartphone as a touch pad to navigate through lists of movie and TV shows on TV and type on an on-screen keyboard. They navigated with and without haptic feedback in a counterbalanced order. They all preferred haptic feedback. The second user study was to navigate on a grid-based TV UI. The errors, task time, number of cells navigated and user ratings were analysed with ANOVA for tracking speed and haptic conditions. The effects of time, errors or ratings were not significant. But eight of the nine participants preferred haptic feedback over no haptic feedback. Thus even there were no performance effects, participants preferred haptic feedback. They conclude that adding haptic feedback significantly improves self-reported satisfaction and that it leads to improved user experience.

2.4 Conclusion

After some research to academic and industrial interaction techniques with TV, we've seen that there are some similarities between them for voice recognition. Gesture-control in industrial examples are different than academic research. The most useful industrial examples are Apple TV and Android TV for voice-search because these use natural language and are not like Amazon Fire TV which doesn't support natural language commands. The industrial examples for gesture-control always use a camera on the TV (e.g. Samsung Smart TV, see Section 2.2.4). If we compare the industrial example Samsung with the Leap Gestures for TV elicitation study, we see that there is a big difference in gestures. The gesture recognition for Samsung is also more like a pointer, where the user moves his hand to a GUI part on the screen for example to put the volume higher and do a long grab on this "button". The initial gesture to start the recognition is "Waving a hand" for Samsung and "Open palm" for Leap gestures (waving a hand was the second choice of the participants of the elicitation study, see Section 2.3.2 to read more about this study). Samsung doesn't make it easier for the user to interact with the TV, it lets the user just interact with a GUI by pointing and grabbing with his/her hand. While with the Leap Motion the user can for example just by drawing the letter "M" go to the Menu, instead of moving his/her hand and grabbing something on the GUI to go to the Menu. The Samsung gesture recognition also can't detect gestures at finger level, which is a drawback and let the users have less gestures to choose of. With the Leap Motion it's possible to detect the hands and fingers better, this lets us have more gestures (see Section 2.3.2).

Apple TV and Android TV use a continuous speech feedback design, this means that the recognized words are shown in real-time while the user speaks and are corrected while speaking (see Section 2.3.4). They both don't show any examples or suggestions on the screen for the users, this enables the users to be free in their search sentences. There should be some boundaries or help for the user to extrapolate sentences from the examples. An example for how to do it can be found in Section 2.3.4.
The two research papers about how participants try to recover their failed recognized sentences (see Section 2.3.5 and 2.3.6) are interesting for my implementation. The two papers are similar to each other, thus the recovery strategies of participants are most of the time the same. We can predict which strategies the users are going to use and we have to implement the voice recognizer to know these strategies of recovery.

If we summarize academic research and industrial examples, Apple TV and Android TV are the most valuable for voice-search. Stifelman et al. [51](see Section 2.3.4) is a helpful guide by designing a voice-search user interface. The two research papers about recognition recovery strategies (see Section 2.3.5 and 2.3.6) are also interesting for voice-search implementation. Leap Gestures for TV (see Section 2.3.2) is interesting for the implementation of gesture-control, especially for the choices of gestures.

Chapter 3

Implementation (Internship)

3.1 Introduction

For my internship at Zappware[1], I got the task to expand their mobile application by implementing two more interaction technologies, voice and gestures. It was important to focus with the voice technology on voice-search and with the gesture technology on navigation tasks. The mobile application connects with a server which receives the voice-search queries and gestures and sends them to the set-top box (STB) which is based on HTML5. The user is able to search by voice using natural language, and the gestures are executed by using a ring. Every command is sent with the help of sockets to the server which actually forwards it to the STB by adding extra information to open the right window or execute the right command.

[1]Zappware, located in Belgium, combines uniquely creativity and technology into powerful digital TV solutions for pay-TV operators exploiting DVB, IPTV, OTT and hybrid networks. Its platform provides an intuitive and personalized multi-screen TV experience across set-top boxes, connected TVs, smartphones, tablets and PCs.

3.2 Internship Assignment

Zappware wants to focus on innovative ways of interaction (find and watch TV content) with mobile devices and TVs with this internship assignment. Currently interaction mainly occurs through touch control (mobile devices) and a physical control (TV). In order to offer a more natural way of interaction combined with ever expanding set of features, this internship targets two fields: voice-control and gesture-control.

Voice-control should be considered as an additional way of interaction. Thus not a method to completely replace the current control schemes (touch and remote control). The starting point will be to find ways of using voice to simplify and enhance certain aspects of the current user experience. A common case is inputting text for a search request (text input). This is currently done through an on-screen keyboard or digit keys on the remote control (SMS-style). Simply saying a search string would greatly enhance the search functionality. This search request should be in natural language and it should be as free as possible. Easy access to different parts of a UI is another important feature of any control scheme (Quick access). In these times of multiple content sources combined with large feature sets, quick access to both screens and individual actions is a major point of interest in any UI. The user just should for example go to the channel list by saying "show me the channel list" instead of going in the menu and looking where the channel list is. I also had the task to assess if voice input can also play a role in the general navigation. For example moving up and down, or tuning to another channel. A major point of attention was how to distinguish regular conversation from voice input intended for the mobile device or TV.

Gesture-control has the potential to become a replacement of the current control- and input methods. Therefore was it very important to investigate the role gestures can play and how precise their input can be. Complete body tracking was out of scope, since the main focus is on interaction with content (from the couch) rather than actively playing a game (standing up). Optional eye tracking can complement gesture-control in refining the input. The points of attentions with gesture-control are how to distinguish: 1) gestures from multiple people and 2) regular gestures and gestures intended for controlling UI. We've used first a Leap Motion to implement gesture-control and later a ring device. It should be possible for the user to navigate in the menu, TV Guide, or just in the UI of the set-top box with gestures. It should also be possible to switch between channels using gestures.

3.3 Architecture

After some research to hardware and software we have decided with Zappware to use a library from a company called SemanticEdge (library can be found in Appendix B.1) for voice-control and a Logbar Ring Zero [18] for gesture-control.
We've used an Android smartphone for the voice-control part because the library SemanticEdge (Appendix B.1) only provided the library for Android. We've added the voice-control to the existing application. We've also used an iPhone smartphone to receive the gestures from the Logbar Ring Zero [18], because there wasn't an application for the Android available nor a SDK. Both applications will send voice and gesture commands/searches to the Node.js server. The server will identify the received messages and add some information and send it to the set-top box. On the figure 3.1 you can find an overview of the architecture.
We'll first explain the voice-control part in Section 3.3.1 and gesture-control in Section 3.3.2.

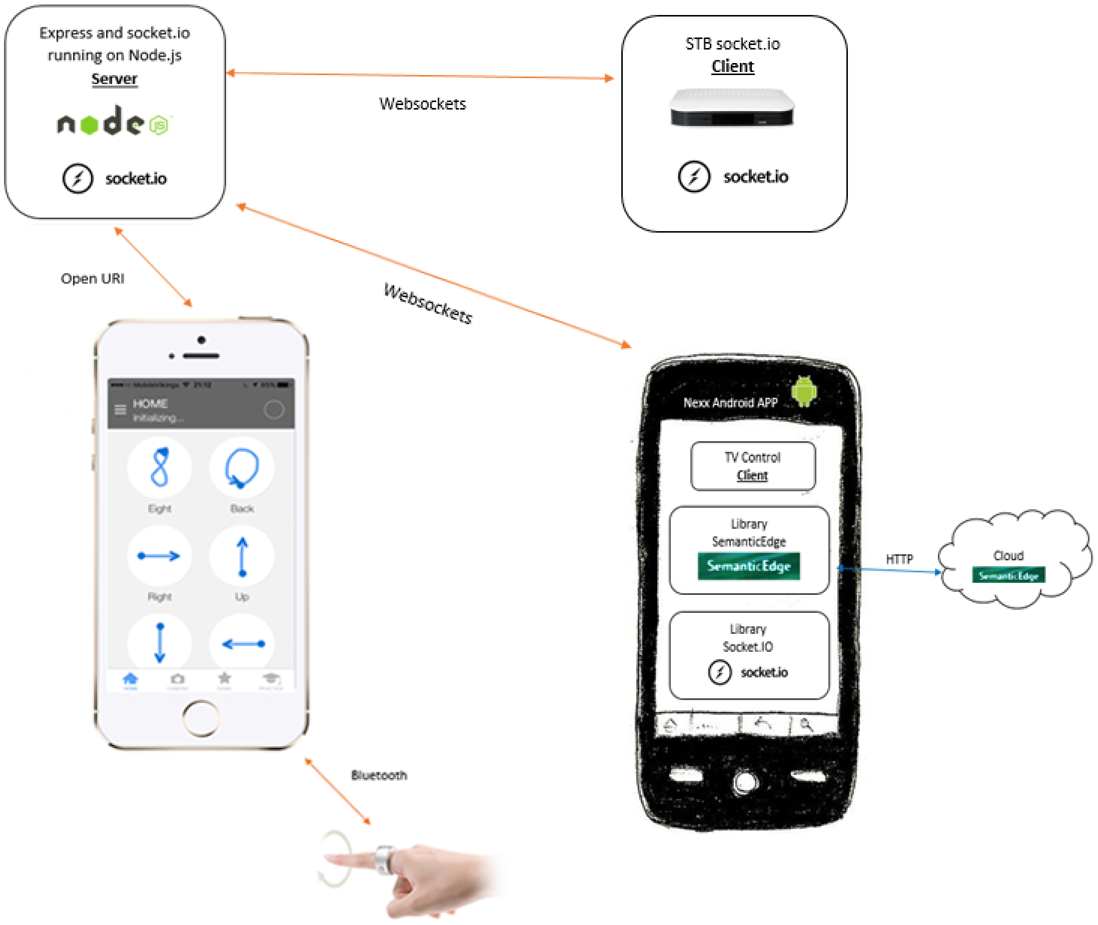


Figure 3.1: The architecture of the implementation

3.3.1 Voice-control

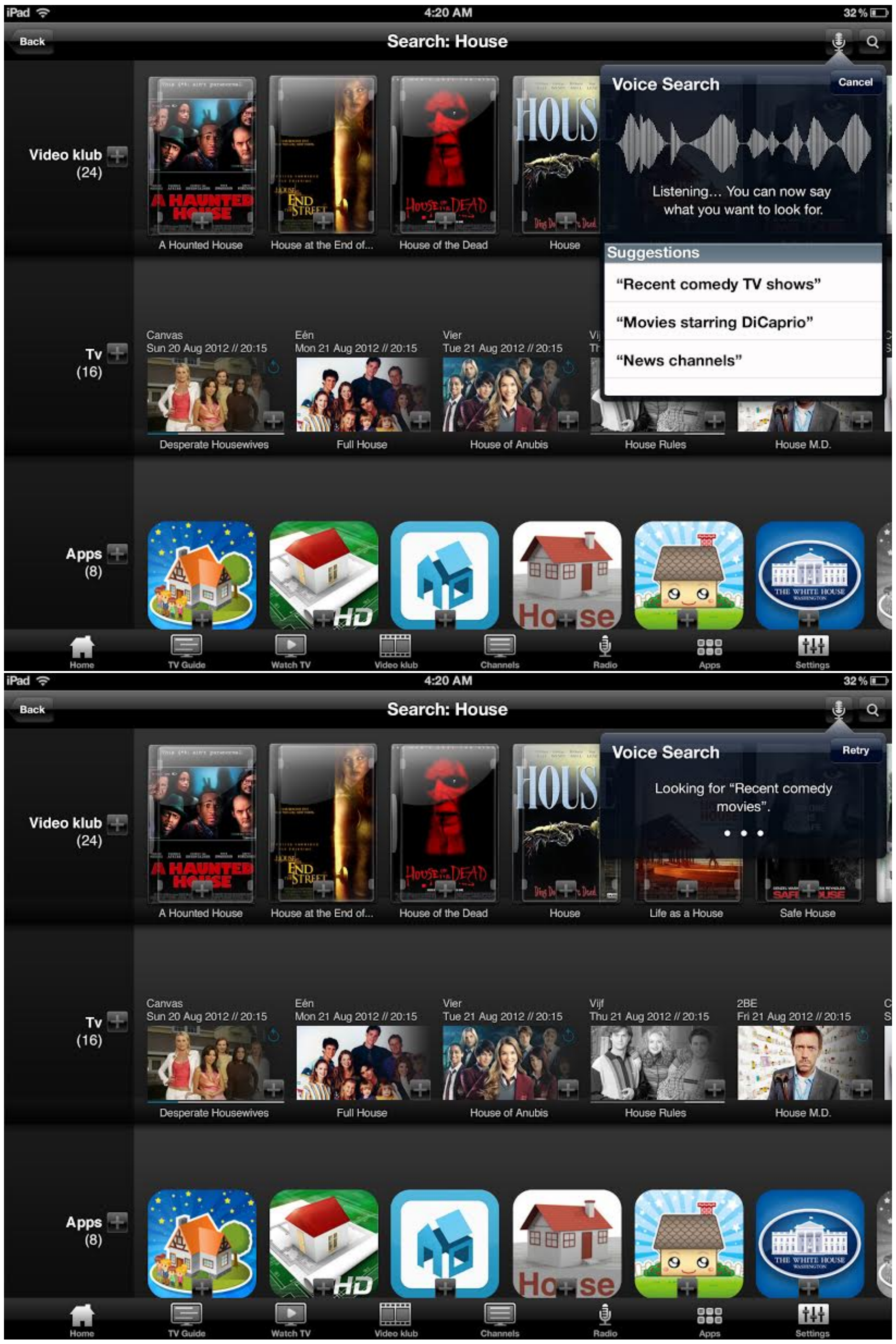
Android application

I first had to examine the existing complex application. The existing application is an Android application to control a set-top box. This application gives an overview of the promotions, and the highlights of TV shows. There is a function to search in the TV Guide and VOD[2] catalogue by typing a search request. It's possible to connect with a set-top box and control it remotely. It has a functionality to watch the channels on the Android device. The user can check the TV guide, the channel list, list for video on demand, watch TV channels live, set reminders,...

One of the important tasks was to implement search functionality by simply saying a search string, this string should be in natural language and free as possible. We've used a cloud-based speech recognizer and semantic analyser called SemanticEdge Dialog client API (see Appendix B.1.3) to recognize voice-search requests and to analyse them. This library offers a complete solution for integrating a multi-model user-interface into existing TV platforms. It provides speech recognition and semantic interpretation in EPG search, Video on demand search, Youtube search and Remote control actions (more information can be found in Appendix B.1). We've also used this library for navigation like tuning to another channel (e.g. "Switch to National Geographic"), and adjusting the volume (e.g. "Volume 20"). We didn't use voice recognition for navigational tasks like moving up/down or left/right in the menu because these navigational tasks have no extra advantage with voice recognition. It's slower to do this with voice recognition than a remote control or gesture-control.

The GUI design of the Android application had to be adapted to the new functionality, voice-control. We have discussed this in our Zappware team, it was important that voice-control was consistent with the other search options. The graphic designer created some assets (see Figure 3.2), as seen on the figure the first asset contains the popup with voice input screen and suggestions. The second asset shows a popup which contains that it's searching for the voice query. These assets were created for IOS devices while we were developing for an Android application. Thus, we've changed the design to adapt it for Android.

[2]Video on demand



Listening... You can now say what you want to look for.

Suggestions

"Recent comedy TV shows"

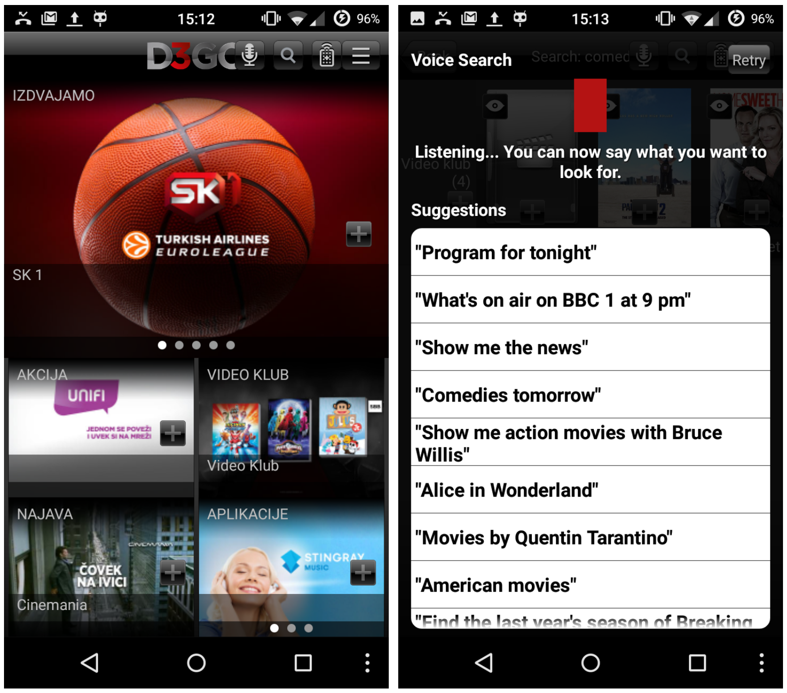
"Movies starring DiCaprio"

"News channels"

Looking for "Recent comedy movies".

Figure 3.2: Voice input and searching screen design

A microphone button is added on the action bar of the Android application, like shown on the asset, which triggers the SemanticEdge Library to recognize the voice query, and shows a pop-up (see Figure 3.3). This pop-up contains a retry button, an animation for the amplitude of the voice and a list of suggestions/examples. This list of examples helps establish boundaries [50].
When the user finishes his/her voice-search the library does a semantic analysis (the functions can be found in the Library Appendix B.1.3), which returns the results in JSON format.
The results are shown in two lists on the application, first list contains the EPG results and the second list the VOD results.



Listening... You can now say what you want to look for.

Suggestions

"Program for tonight"

"What's on air on BBC 1 at 9 pm"

"Show me the news"

"Comedies tomorrow"

"Show me action movies with Bruce Willis"

"Alice in Wonderland"

"Movies by Quentin Tarantino"

"American movies"

"Find the last year's season of Breaking

Figure 3.3: The existing mobile application with microphone button added (left) and with the voice popup(right)

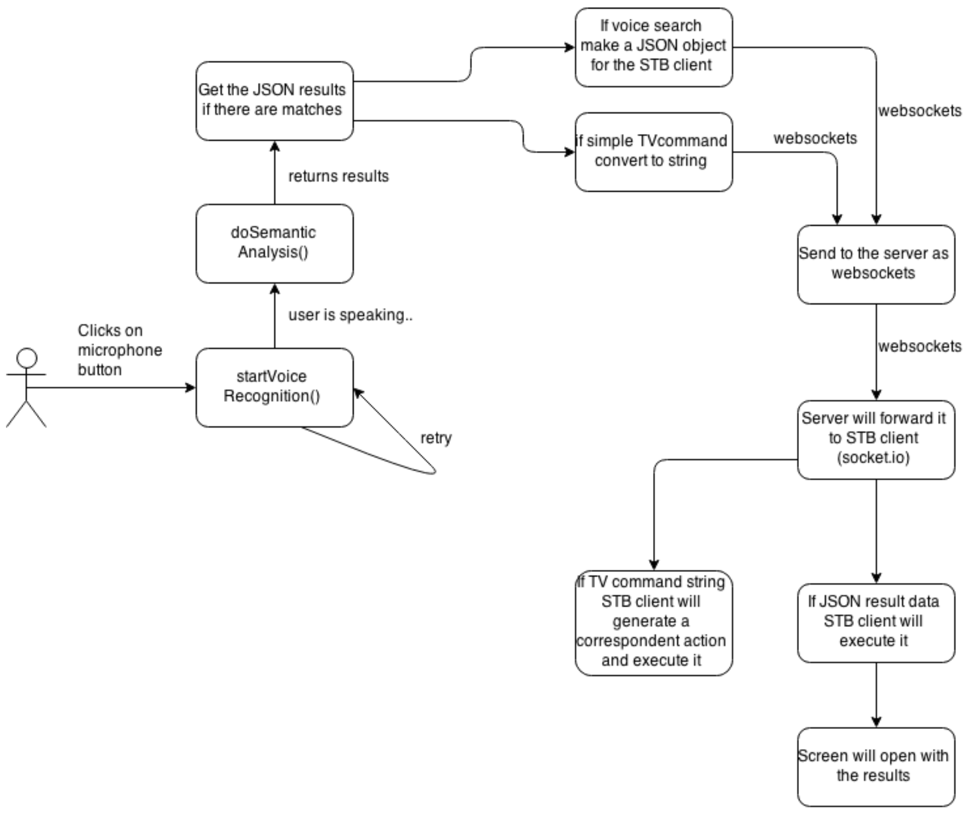
SemanticEdge The voice recognition and semantic analyse was implemented with SemanticEdge Dialog client API (Appendix B.1.3). A SemanticEdgeHandler class was implemented, this class will handle the connection, voice recognition, semantic analyse and error handling (e.g. No Network, No speech found, No match, ...). In this class the results are checked separately as a TVCommand or a VoiceSearch. If it's a TVCommand, it checks the JSON object we get from SemanticEdge and converts it to a String with the action included (e.g "VolumeUp"). If it's a VoiceSearch it converts the JSON object from SemanticEdge to a JSON object which will be accepted by the set-top box. This JSON object will have the structure of a dialog with the search results included.

SocketIO The connection between the Android device and the NodeJS server is implemented with the library Socket.IO v1.x Client Library (see Appendix B.3). A SocketIOHandler class is implemented which establishes the connection and sends/receives websockets. The user can set the IP of the server that is needed to be connected in the settings of the Android application. After these settings have been set the Android device will be connected to the server, and it can now send and receive websockets. There are two functions to send the commands: sendJSONMessage(JSONObject json) and sendTVCommand(String command). The first function sends a JSON object with the results and the right structure to fill in a window on the set-top box, the second function contains the voice TV command as a String (e.g. "VolumeUp" or "Channel 45").

Program flow

In the following diagram (see Figure 3.4) the program flow is shown. The user clicks the microphone button on the Android application, this will trigger the SemanticEdge library to start the voice recognition. The library will listen for a couple of seconds and it also detects when a user stops speaking. This will return the recognized text (if it has recognized what the user said otherwise it will show "no match found" or "no speech found"), then it will do a semantic analysis on this string. Thus, the library is going to look if there is a match on the cloud. This will return a JSON object if there is a match with the data on the cloud. There are two types of data, TV command data or monitor content data (voice-search). If it is a monitor content data, it will convert it to another JSON object which is readable and executable by the STB client. It will also show the results on the application as a catalogue of media items. If it's a TV command data it'll convert it to a string. The simple TV commands that have a correspondent action on the STB client are, "VolumeUp", "VolumeDown", "ChannelUp", "ChannelDown", "Channel *nr*".
This JSON object or string will be sent through websockets (Socket.io see

Appendix B.3) to the server. The server will then forward these sockets to the STB client. The STB client will make a distinction between the voice-search and TV command. If it's a voice-search JSON object, it will just execute it and a result screen will be shown. Otherwise if it's a TV command string (e.g. "VolumeUp"), it will generate a JSON object which contains the action to do the command.



Clicks on microphone button
startVoice Recognition()
retry
user is speaking..
doSemantic Analysis()
returns results
Get the JSON results if there are matches
If voice search make a JSON object for the STB client
If simple TVcommand convert to string
websockets
websockets
Send to the server as websockets
websockets
Server will forward it to STB client (socket.io)
If TV command string STB client will generate a correspondent action and execute it
If JSON result data STB client will execute it
Screen will open with the results

Figure 3.4: Program flow

3.3.2 Gesture-control

First, we've used a Leap Motion to implement gesture-control. This is done because the ring device was not available at the time and to investigate how precise the input is with a Leap Motion. When the Logbar Ring Zero [18] was available, we implemented gesture-control with this device. With the Leap Motion the user has nothing to wear and can do 3-D gestures with one or both hands. But by using a ring, the user is wearing a device on his/her index finger and can only do 2-D gestures with this finger. Thus, with a Leap Motion the user is freer and can do more gestures than a ring device, but according to Stifelman et al. [51] participants prefer 2-D gestures. We'll first describe the implementation with Leap Motion and later the implementation with Logbar Ring Zero.

Leap Motion

We implemented gesture-control with Leap Motion on the set-top box in the programming language JavaScript (Leapmotionhandler class). Because the whole body tracking was out of scope we only had to focus on interaction with content from the couch and not standing up, therefore Leap Motion was a good choice. We implemented some gestures such as numbers to switch channels, putting volume higher and lower, switching channels up and down.
For putting the volume higher and lower, we chose a gesture up and down with one hand grabbed. The higher the user's hand moves the higher the volume will be, because of Leap Motion it is possible to give direct feedback to the set-top box. Thus, we'll not wait until the gesture is completed but we're putting the volume up or down while the user is moving his/her hand grabbed up or down (see Figure 3.5). This creates a direct-feedback where the user has full control of the volume. This cannot be done with the ring, because it's impossible to get feedback while doing a gesture with the ring. There is only an after-gesture feedback when we are using the Logbar Ring Zero device.

The gesture to switch to the next channel and previous channel are the same as the Leap Gestures for TV elicitation study (see Section 2.3.2). To switch to the next channel we've used a hand gesture (not grabbed) from left to right. And to switch to the previous channel we've used the opposite, from right to left. These gestures are the most common for switching channels, and are used widely.

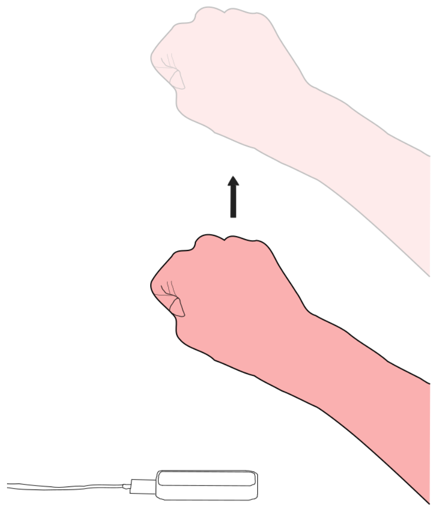


Figure 3.5: Leap Motion: Volume hand gesture

To switch to a specific channel we've implemented it with number gestures, where the user draws a number in mid-air. These gestures are also similar to the Leap Gestures for TV elicitation study (see Section 2.3.2).

The Leap Motion is connected with USB to a Laptop where the set-top box is running on HTML5 (see Section 3.3.4), and the Leap Motion Controller is installed on the laptop. Therefore there is no need to have a connection with a server, the communication goes directly with the set-top box.

Logbar Ring Zero

We also implemented gesture-based interaction using a Logbar Ring Zero [18] after implementing it with a Leap Motion. The Logbar Ring Zero is the smallest input device according to them. This device has a touch sensor, which triggers the recognizer. The recognizer will only recognize while the user is touching the sensor. When the user releases the touch sensor, it should recognized a gesture. The user has to put this ring on his/her index finger, and use his/her thumb to touch the touch sensor (see Figure 3.6). The ring also gives feedback by vibration when the gesture is recognized as one of the listed gestures or when it's not. Thus, there are two different vibrations one for recognized correctly and one for recognized wrong. The

Logbar Ring Zero can only recognize 2-D gestures. The ring is connected to

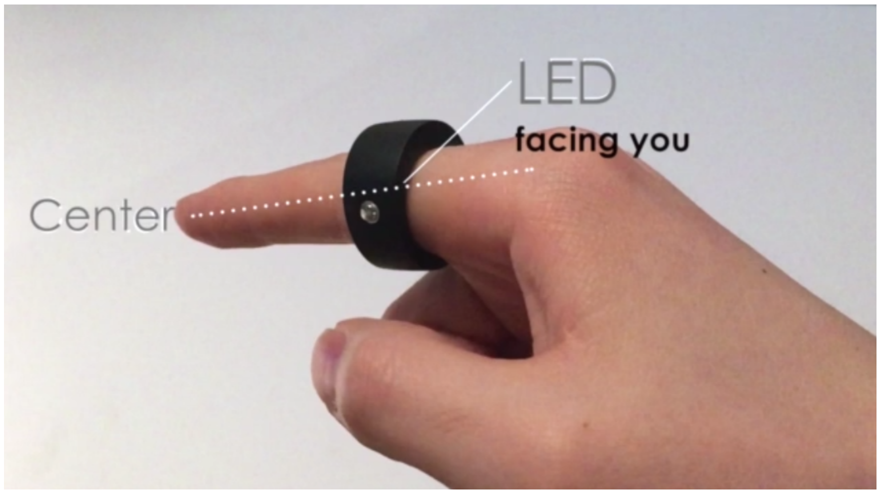


Figure 3.6: Logbar Ring Zero [18]

the Ring Logbar application on an iPhone with Bluetooth (there is not yet an application for other platforms or a SDK). With this application it's possible to add new gestures and customize them. You can easily add gestures by drawing the gesture using the touch screen of your smartphone, and save it. The 2-D gesture of the user in the air will be mapped with the one saved in the application.
We've added 20 gestures to the application which are described in Table 3.1 and Table 3.2. Every gesture has an Open URI assigned, this will be recognized by the Node.js server.

Gestures

The choice of gestures with which one will interact with the TV is important. The gestures have to be 2-D gestures, thus we have only focused on these types of gestures. We implemented gestures for: quick access to the menu, to select an item, to go back, to switch between next and previous channel, to navigate left, right, up and down and to switch between channels with numbers. In the Tables 3.1-3.2, you can find all the gestures.

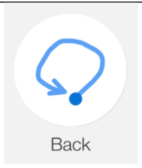
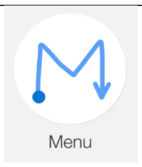
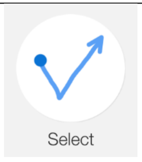
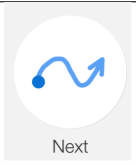
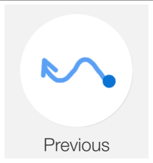
Name	Gestures
Back	
Menu	
Select	
Next and previous channel	 

Table 3.1: Gesture list

Some of my gestures are based on Leap Gestures for TV elicitation study [55], discussed in Section 2.3.2. In this study they used a Leap Motion which can also recognize 3-D gestures, but the users preferred 2-D gestures over 3-D. Which is good because we're using a input device which can only detect 2-D gestures. The menu gesture for example is from this study, they concluded that users also prefer to draw the first letter of task names in mid-air to execute it.
We've chosen an "undo" gesture to go back and a check mark gesture to

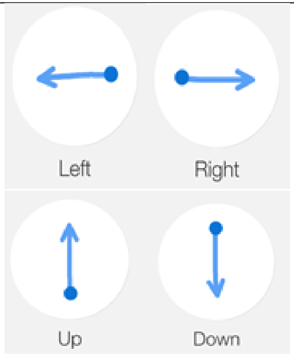
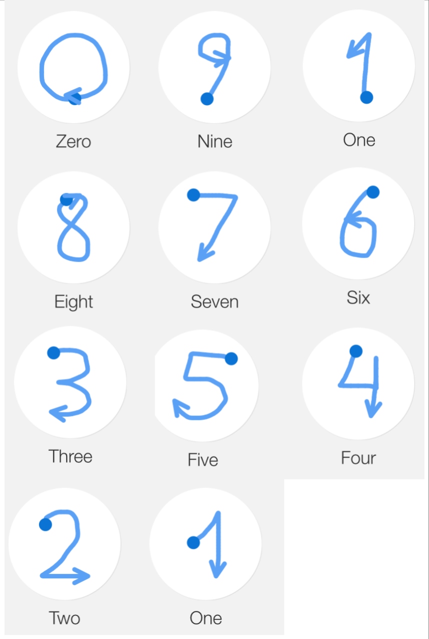
Name	Gestures
Navigation: left, right, up and down	
Numbers (0-9)	 Zero Nine One Eight Seven Six Three Five Four Two One

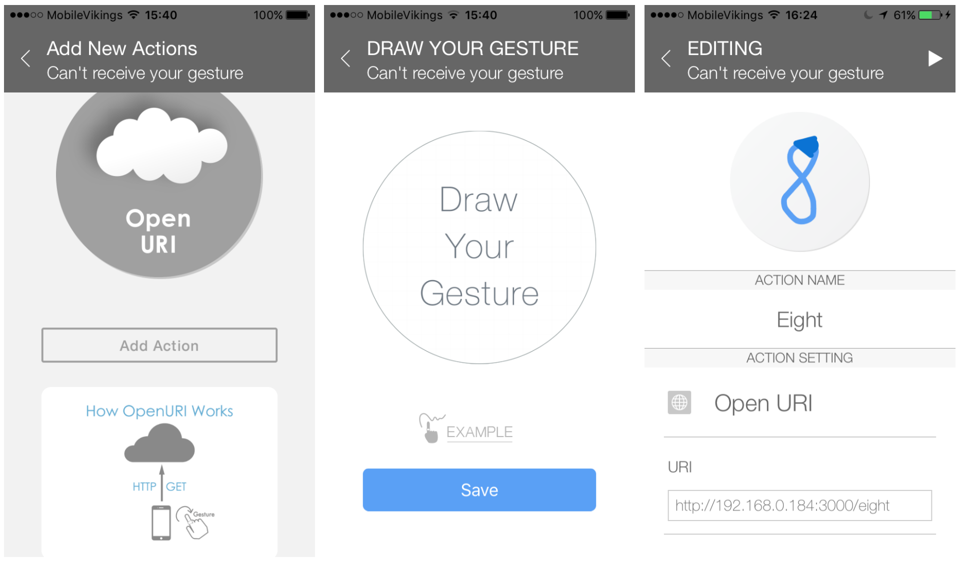
Table 3.2: Gesture list

select. The navigation gestures are basic gestures, which are always the same in most of the studies. To go to a next or previous channel we've used gestures similar to left and right but still different. It was important that the gestures were different enough of each other for the ring to recognize the right gesture. It was also important to know how users write numbers and letters, we've implemented the number gestures in every possible way to write it in the air. For example it's possible to write the gesture "1" or

"8" in two ways. Both will be recognized as the same gesture.

Application

The Logbar application on IOS is used because there was no SDK for developers. This was the only way to make my own gestures for the application and let them recognize and be sent to the Node.js server. On the application side we've configured the gestures by choosing our own action, It's possible to draw everything in 2D and we had to assign an Open URI to every gesture (see Figure 3.7). Every gesture has an OpenURI which is in this form "http://IP:PORT/GESTURE". It's possible to give multiple gestures the same URI. This is useful for the number gestures, because some numbers can be drawn in multiple ways. The Logbar application should not always shown on the screen, it can also run on the background.



●●●○○ MobileVikings 15:40 100%
Add New Actions
Can't receive your gesture
Open URI
Add Action
How OpenURI Works
HTTP
GET
Gesture
●●●○○ MobileVikings 15:40 100%
DRAW YOUR GESTURE
Can't receive your gesture
Draw Your Gesture
EXAMPLE
Save
●●●●○ MobileVikings 16:24 61%
EDITING
Can't receive your gesture

ACTION NAME
Eight
ACTION SETTING
Open URI
URI
http://192.168.0.184:3000/eight

Figure 3.7: Logbar Ring Zero app [18]

Program flow

The user does a gesture with the Ring, the Logbar application running on the iPhone recognizes the gesture and gives feedback to the user (vibrate) and opens the corresponding Open URI link of the gesture. The Node.js server does a get on the corresponding gesture and creates a socket with the gesture action included. The server forwards this socket to the set-top box Node.js client which only accepts sockets of the type Socket.io. The set-top box client will now check which gesture it is and do the correspondent action.

3.3.3 Node.js Server

Zappware wanted a connection with a Node.js server which could send and receive websockets. After some research we've chosen to use SocketIO (see Appendix B.3), it is a JavaScript library for real-time web applications. It's possible to have a bidirectional communication between the clients and the server and it's real-time (fast communication). It enables real-time, bidirectional communication between web clients and a server. It consists of two parts, one part is the client-side library that runs in the browser, and a server-side library for Node.js.

We've implemented a server with Node.js (see Appendix B.2). The server is running on a local host on Node.js. This server will receive messages from the Android application (see Section 3.3.1) and Logbar application (iPhone) (see Section 3.3.2), and sends it to the set-top box. The server consists of two parts, the first part will only accept voice-searches or commands and the second part will accept gestures.
The messages sent and received from voice-search are SocketIO sockets (see Appendix B.3). For the gestures messages we've used the express [16] module to receive, this is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications, and Socket.IO to send the gesture action to the set-top box.

For the voice-search part of the server, we've made a distinction between voice-search results and voice TV commands. The first are sent as a JSON Object from the Android application, the second are sent as String. The server is going to receive them both separately as followed:

```
socket.on('messageFromAndroid', function (content) {
    if(hnexx_socket != null)
        hnexx_socket.emit('message', content);
});

socket.on('TVcommandFromAndroid', function (content) {
    if(hnexx_socket != null)
        hnexx_socket.emit('TVcommandFromAndroid', content);
});
```

The first "messageFromAndroid" contains the voice-search results with the attributes needed to create a window with the results on the set-top box. "TVcommandFromAndroid" contains only a voice command (e.g. "VolumeUp").

The gestures are received by the module Express, basically every gesture has an OpenURI as stated before of the form "http://IP:PORT/GESTURE". At the server side we'll get the gesture and send them to the set-top box as a TV Command of the type String. For example if the gesture we want to receive is the number one, to switch to channel 1. The OpenURI of gesture will be "http://IP:PORT/one" and at the server side we'll receive it as followed:

```
app.get('/one', function (req, res) {
  sendkey(125, function() {
    res.sendStatus(200);
    console.log("1");
    if(hnexx_socket != null)
     hnexx_socket.emit('gesture', "Channel 1");
  });
});
```

This code will get the gesture to switch to channel one. If the set-top box is not null, thus if there is a set-top box connected it will send the action to the set-top box with Socket.io.

3.3.4 Set-top box

The set-top box has a HTML5 based interface. We've added the channels that are supported by the Android application (SemanticEdge library) first at the same order as the Android application. This should been done because when we want to switch to a channel by voice for example "switch to National Geographic", this channel should have the same channel number on the Android app and the set-top box. The graphic designer of Zappware has provided the channel logo's.

We've created a new handler called "websockethandler.js" which connects with the Node.js server when the set-top box is on. We'll listen to the three incoming sockets: 1) a voice-search with the JSON included to open a screen with results, 2) a voice command (channel *number*, channelUp/Down, volumeUp/Down/Mute), 3) a gesture (channel *number*, left/right/up/down, menu, back, select, next/previous channel). If it's a JSON object we'll just execute it because it already contains all the information to open a screen with the voice-search results. If it's just a string with a simple TV command or a gesture, we'll generate a JSON object which contains the corresponding action and execute this.

On Figure 3.8 below the screen of voice-search results are shown of the voice query "Show me the news". This list will only contain results of the EPG Guide, but when there are also matching results in the VOD catalogue there will be two lists one with EPG Guide results and another with VOD results.

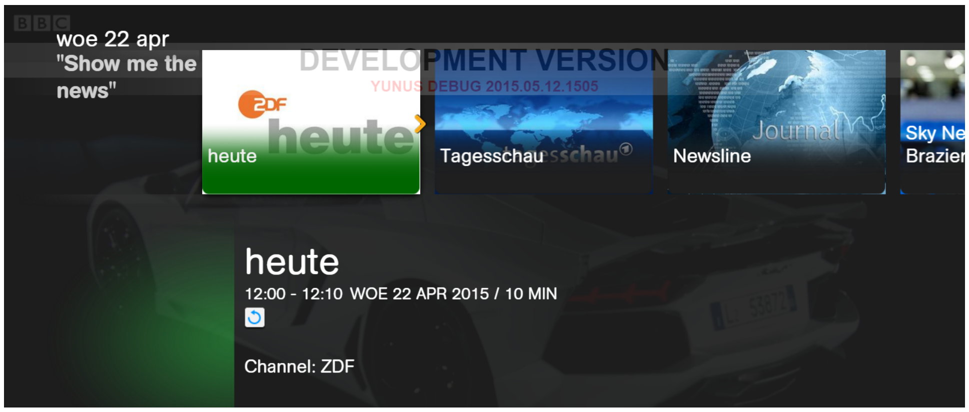


Figure 3.8: Set-top box UI with voice-search results

3.4 Conclusion

Implementing the gesture-control with two different input devices made me able to make some differences. While with the Leap Motion it is possible to have more gestures and support two hands, Logbar Ring Zero only supports the index finger (less gestures). Leap motion supports 3-D gestures, the Logbar Ring Zero can only detect 2-D gestures. While the Leap Motion seems to be in advantage, Logbar Ring Zero has more possibilities for gesture-control with a TV.

The points of attention for gesture-control were how to distinguish gestures from multiple people and how to distinguish regular gestures and gestures intended for controlling the UI. By using a Logbar Ring Zero, there is probably one (or more) rings connected and every user has his/her own ring. So the distinction is really easy, it's also probably possible to know which gesture is from which ring by the ID of the ring. It's easy to distinguish normal gestures and gestures with the intent to control the UI with the Logbar Ring Zero. Because when the user wants to do a gesture to control the UI. He/she has to touch the touch sensor while doing the gesture, by having this touch sensor we can distinguish normal gestures and gestures to control the UI.
If we are using a Leap Motion it's easy to distinguish gestures from multiple people because the range of the device is not so high [24]: x = 117,5 mm, y = 317.5 mm and z = 73.5 mm. Therefore only one person's hands will be recognized while sitting on the couch. To distinguish normal gestures from gestures with the intent to control the UI, we can implement a gesture that triggers the gesture-control (e.g. "Wave" gesture like the Samsung gesture-control, see Section 2.2.4).

One disadvantage of the Leap Motion is that it should be placed close to the user, but it's connected with a cable so it will be wired. And when another user wants to control the TV it has to be moved or the user has to sit close to the Leap Motion to be able to do gestures. When we are using a Logbar Ring Zero there isn't a disadvantage, the connection is with Bluetooth and it's easy to handover the ring to a second user or we can just use multiple rings.
The initial setup was to use a Logbar Ring Zero and because it's more interesting and there are no user studies about it yet, we'll use a Logbar Ring Zero instead of a Leap Motion in the user study combined with voice-control.

In the implementation of voice-control it's important to use a speech recognizer that is specialized in the domain you want to use it. We have used the SemanticEdge library which offers a complete solution for integrating a multi-model user interface into existing TV and video platforms. Voice-control had to be implemented as an additional way of interaction, firstly one of the most important functions of voice-control was voice-search. Voice-search is implemented with the SemanticEdge library which searches in the EPG guide and VOD catalogue. Quick access to channels by just saying the channel name is implemented with the same library. It was important to have the same channel list at the application side (speech-recognition side) and at the set-top box side for the library. Implementing voice-control for navigational tasks like "moving up/down" is worthless if there is gesture-control and a remote control. But voice commands like "Volume 30" are more interesting for voice-control, because this cannot be done with a remote control in one time. The voice-search was added to an existing Android application, the microphone of this smartphone is used to recognize the speech therefore background noise will not be a big problem. Also the distinction between regular conversation and voice intended for the mobile device was important, we've done this by putting a button to start the voice recognition and the application analyses the voice sentence when the user stops speaking.

Chapter 4

Evaluation

4.1 Introduction

We're interested in understanding people's preferences for interacting with TV with this user study. We're trying to understand how they feel using those interaction methods alone and in presence of others, what they think about the future of these interaction methods. We're validating the value and intuitiveness of gestures set, and the voice recognition.
We're interested what the participants think after the experiment, if they have changed their mind after using my application. And if they would switch to a system like this, and use it at home.

4.2 Method

4.2.1 Participants

Fifteen (15) volunteers (1 female) participated in the study (age range 19-25 years). One participant was left-handed and the rest was right-handed. One participant was a worker, the rest were students. Three participants were using the voice interaction application on their smartphone. Five participants had no previous experience with gestural interfaces, whereas the other ten people had used Nintendo Wii and PlayStation Move for games, and Leap Motion (Computer Science students). All the participants owned touch-screen smartphones.

4.2.2 Apparatus

A 40-inch (102 cm) Phillips Smart TV was connected to a laptop (with HDMI cable) running on Microsoft Windows 8.1. The Logbar Ring Zero (see Section 3.3.2) was connected with Bluetooth to an iPhone to recognize gestures and send them to a Nodejs server running on the laptop. An Android phone was used for voice recognition and understanding, this was also send to the Node.js server running on the laptop. The Node.js server was sending the incoming gestures and/or voice-search/commands to a HTML5 TV UI that was showing on the TV.

4.2.3 Procedure

Before running the study, participants were asked first to fill in a survey about their background and their use of voice and gesture interaction (this survey can be found in the Appendix C.1). During the experiment the participants were seated comfortable at approximately 2 meters from the TV set. The experimenter was present during the entire duration of the study with the role to introduce participants to the features of the Logbar Ring Zero and to the Voice recognition application.
The tasks were split into two parts, the first part was the voice recognition (natural language understanding) part and the second part was the gesture interaction part (with the Logbar Ring Zero).

Voice Recognition

The participants were asked to search for: movies of the genre comedy, movies with a specific actor, a specific movie, football programs on tv, programs about cooking this week, programs on a specific channel, only American movies and to search something of their own choice. The participant was also asked to find a voice command to switch to channel BBC World News and to put the volume higher.
At the end of the experiment the participants were asked to fill in a questionnaire (Appendix C.3) in which they have rated the easiness to find the search string on a 5-point Likert scale, with 1 denoting "very easy" and 5 "very hard". And the questionnaire also included a 5-point Likert scale, with 1 denoting "never" and 5 "always" for the following questions: Did your search strings matched what you were looking for? Did you expected the search strings would give the results? And would you use the voice recognition at home?

Gesture Recognition

Before running the tasks of gesture recognition, participants were given time to familiarize with the Logbar Ring Zero. Participants were asked to play a tutorial game of 60 seconds with the Logbar Ring Zero to get used to it. The participants were asked, after the training, to do the following actions:

- Navigate to the menu
- Navigate to the list of channels,
- Navigate to channel number 3 in the list
- Navigate to channel 8 by drawing the number
- Navigate to a channel of your own choice by drawing
- Switch between channels with the next and previous gestures
- Change the interface language to French

The participant first had to think about a fitting gesture and could check the list of gestures afterwards which contained all the possible gestures. After the experiment the participants rated how they expected the navigation gestures would work, how it actually worked, the intuitiveness of the gestures and if they would use it at home in a questionnaire on a 5-point Likert scale. The participants also gave scores on 10 for each gesture and they had the opportunity to write their own gesture instead (see full survey on Appendix C.4).

System Evaluation

After the experiments the participants were asked to rate the system first with the Single Ease Question (SEQ), this is a 7-point rating scale to assess how difficult users find a task. SEQ performed about as well or better than more complicated measures [41]. There is also a correlation (r=0.5) between the user responses and the task-time and task-completion. Users are more likely to rate tasks more difficult if it takes longer to complete it or if they don't succeed at all.

They were also asked to rate the system with the System Usability Scale (SUS), this is a 10 item questionnaire with five response options; from strongly agree to strongly disagree (see Appendix C.5).

4.3 Results

4.3.1 Voice Recognition

Survey before the experiment

In the questionnaire before the experiment (see Appendix C.1) we've asked what the participants feel when they are using voice recognition and what they think about the future of it. Five participants were feeling awkward when they were using voice recognition, two participants were feeling frustrated because in their experience the application they have used was not properly recognizing the voice commands, one participant was feeling rich and saw it as a luxury, another participant was feeling ashamed and would never use it in public, the rest of the participants felt nothing or hadn't used it yet. Also only one participant was using voice recognition in presence of others. The reasons what makes voice interactions awkward and embarrassing according to Sharma [49] are that the voice recognition is imperfect thus the possibilities for errors are common. Second, the progression from personal computers to smartphones/tablets has made computing devices very personal, they offer more privacy by using touch UI to interact on their smaller screens. Thirdly, voice interaction co-evolved with social interaction, thus voice is an inherently social human-to-human interaction medium [44].
Five participants didn't expect much of voice recognition, four participants saw a future if the computers also can understand the meaning and think, three participants see the future of voice recognitions only for users with disabilities, the other participants think that it will improve.

Experiment

The sequences of utterances by participants of the voice recognition tasks can be found in the Table 4.1-4.2. The participants rated how they did found the search string on a Likert scale, the average result of the participants was that it was very easy or easy to find a matching search string. Errors only occurred when the content the users were searching for was not included in the VOD catalogue. In the Table 4.1-4.2 we can find that there are two types of users. The first category used long sentences to interact with the voice recognition, the second category only used keywords. When we asked why they only used keywords most of them answered that they were too lazy to use a longer sentence and that it was enough to let the application understand.

According to Stifelman et al. [51] there are four cases of impact of the examples on the speech queries. The first is extrapolated from an example (e.g. the example is “Show me comedy movies” and the participant says “Show me action movies”). The second case is repeated verbatim, in this case matches the example exactly with what the participants says. The third case is phrased-uniquely from an example, with a different carrier phrase or content. And the last one is a keyword-style input.

This is the list of examples on the application:“Program for tonight”, “What’s on air on BBC 1 at 9 pm”, “Show me the news”, “Comedies tomorrow”, “Show me action movies with Bruce Willis”, “Alice in Wonderland”, “Movies by Quentin Tarantino”, “American movies”, “Find the last year’s season of Breaking Bad”, “Find Breaking Bad season 5 episode 1”, “Switch to BBC One”.

And this is the list of tasks that can be found in Appendix C.2: “movies of genre comedy”, “movies with a specific actor”, “a specific movie”, “football programs on TV”, “programs about cooking this week”, “programs on a specific channel”, “only American movies”. If we look at the table of sequences of utterances (Table 4.1-4.2) we can see that the examples influenced the input, but they are not restricted to them. As shown in Figure 4.1, we found that only 8,57% of the speech input was comprised of verbatim repeats of examples, 47,62% of the queries are extrapolated from the examples, 17,14% of the time the participants used an unique query. Finally, 26,67% were keyword inputs. We can conclude that examples and/or suggestions have a great impact on the queries created by the users. Participants have extrapolated 47,62% and 8,57% of the queries were exactly matching queries, even though most of the tasks were not able to be created from the list of query suggestions on the application. 26,67% used keywords, according to one participant it was because he was too lazy to make a long query.
Providing a list of suggestions or examples is important for the users to establish some boundaries, but it could also have a negative impact like users parroting examples or perceiving limitations that don’t exist. Stifelman et al. [51] designed a cloud of examples which shows examples with blanks (e.g. “Show me something ____”). Using a cloud of examples will represent each example as many thousands of possible phrases rather than just one. It will help the user to create more extrapolated phrases.

Most of the participants were impressed how good it matched what they

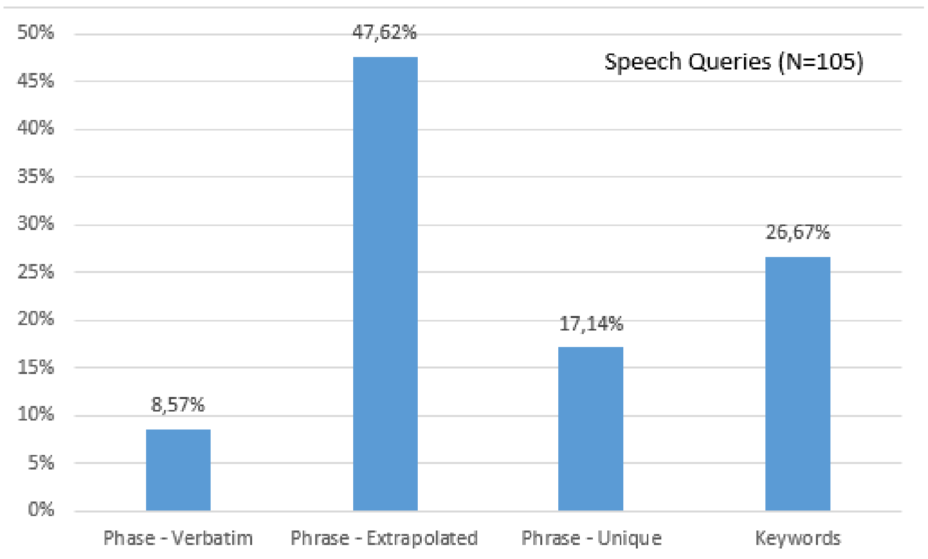


Figure 4.1: Influence of examples on speech input

were looking for, this was most of the time because they had bad experiences with voice interaction and were not expecting much of this method. They were impressed that it could understand not only simple standard voice commands, but that it also could understand natural language. And that they didn't need an English accent to use the system, because the library used for speech recognition is implemented for non-native English speakers.

On the survey before the experiment, we asked if the participants would use voice interaction to search TV content and nine of them have chosen to use it, the other six participants would not. After the experiment we asked if they would use this system at home, eight participants who said previously that they would use this system would use it often, and two participants with the same choice would use it "neutral" (more than sometimes less than often). One participant, who previously said that he wouldn't use such an application, would use it often after the experiment. Another participant would use it also "neutral", and three others would use it sometimes.
Even the participants who had chosen not to use a system like this, has changed their mind. This is because most of the participants were only expecting that the voice-search command would match sometimes or neutral, and it actually matched often or always.

Table 4.1: Sequences of utterances by participants

	movies of the genre comedy	**movies with a specific actor**	**a specific movie**	**football programs on tv**
P1	"Movies of comedy"	"Movies with Jake Gyllenhaal"	"Donnie Darko"	"Show me all soccer programs on TV"
P2	"Comedy movies"	"Movies with Jim Carrey"	"Entourage"	"Football programs"
P3	"Comedy movies"	"Johnny Depp"	"Terminator"	"Football"
P4	"Comedy movies"	"Show me movies with Daniel Radcliffe"	"Need for speed"	"Show me football"
P5	"Show me comedy movies"	"Show me movies with Tom Cruise"	"Show me the Matrix"	"Show me some football"
P6	"Movies of the genre comedy"	"Movies with Sean Connery"	"The movie interstellar"	"Show me all football programs on TV"
P7	"Comedies tomorrow"	"A movie with Morgan Freeman"	"The Dark Knight"	"Football programs tomorrow"
P8	"Genre Comedy"	"Jim Carrey"	"The Dark Knight"	"Football programs"
P9	"Comedy"	"Jason Statham"	"Interstellar"	"Footbal"
P10	"Comedy movies"	"Movies with Johnny Depp"	"Alice in Wonderland"	"Show football programs"
P11	"Comedies"	"Emillia Clarke"	"Game of Thrones"	"Football programs"
P12	"Comedy movies"	"Movies with Morgan Freeman"	"Star Wars"	"Football programs"
P13	"Comedy"	"Emma Watson"	"Harry Potter"	"Football"
P14	"Comedy movies"	"Movies with Al Pacino"	"Pirates of the Caribbean"	"Show me football"
P15	"Show me comedy movies"	"Show me movies with Jim Carrey"	"Show me the Godfather"	"Show me all football programs"

Table 4.2: Sequences of utterances by participants

	programs about cooking this week	**programs on a specific channel**	**only American movies**
P1	“Cooking programs this week”	“Show me all programs on National Geographic”	“Show me only American movies”
P2	“Cooking programs”	“Programs on BBC”	“American movies”
P3	“Show me all cooking programs”	“Show me National Geographic”	“Show me all American movies”
P4	“Show me cooking shows this week”	“Show me programs on BBC One”	“Show me American movies”
P5	“Show me a cooking program from this week”	“Show me programs on National Geographic”	“Show me American movies”
P6	“Programs about cooking”	“Programs on MTV”	“Show me all American movies”
P7	“Cooking programs this week”	“Shows on the BBC One”	“Movies in the US”
P8	“Cooking programs”	“Programs on BBC”	“American movies”
P9	“Cooking”	“Program on Nat Geo”	“American movies”
P10	“Cooking programs this week”	“Programs on BBC One”	“American movies”
P11	“Cooking programs”	“Programs on MTV”	“American movies”
P12	“Cooking this week”	“Programs on Disney Channel”	“Only American movies”’
P13	“Cooking”	“Programs BBC”	“American”
P14	“Show me cooking”	“Show programs on BBC”	“Show American Movies”
P15	“Show me cooking programs this week”	“Show me all programs on BBC World News”	“Show me only American movies”

4.3.2 Gesture Recognition

Survey before the experiment

Most of the participants were feeling nothing at all when they were using gesture recognition in their previous experiences. One participant said “feels way better than voice, it feels more natural”, another participant said that he gets tired after a while.
Most of the participants saw the future of gesture recognition bright, and they think that it will improve a lot and would be used more. One participant thought that “voice recognition has a higher chance to be successful”.

Experiment

Participants used a 5-point Likert scale to rate how they expected the gesture recognition would work. Six participants expected the navigation gestures would work well, five participants expected it would work neutral and four expected that it would work bad. The next question was how it actually worked, eight participants thought it worked neutral, six participants thought it worked well and one participant said it worked bad. Thus it actually worked better than most of the users expected, but still there were a lot of users thinking that it worked neutral. This could be because five participants used gesture recognition for the first time and that they had not enough time to learn.

Gesture goodness Participants used a 5-point Likert scale to rate how intuitive the gestures were, with 1 denoting “very bad”, 2 “bad”, 3 “neutral”, 4 “good”, 5 “very good”. Overall, the mean rating was 4 showing a good intuitive rate and the standard deviation was 0.65465.
The participants were also asked to rate each gesture and they had the opportunity to suggest a gesture. The first gesture was the back gesture, the average score given by participants is 7.9 on this gesture. One participant suggested to use a “X” gesture, to go back. The second gesture was the menu gesture, the average score given by participants is 8.9. Most of the participants liked this gesture, only one participant suggested a simpler gesture because this gesture could be used frequently by users. The third gesture was the select gesture, this gesture had an average score of 8.3. One participant has suggested a point gesture, but this was not possible with the hardware to recognize. The fourth gestures were the next and previous channel switching gestures, these gestures had an average score of 7.8. One participant had suggested to use a double right or left gesture for channel

switching. The next gestures were the navigation gestures (right, left, up, down), the average score of these gestures given by the participants is 9.7. The participants had no suggestion about these gestures. The last gestures were the number gestures (0-9), the average score of this 9.1. The suggestion of the participants were that they drew some numbers different. A comment two participants gave was that it would be nice if they could import their own gestures and assign them to commands. On Figure 4.2, you can find a graph which shows the gesture goodness. We can clearly see that most of the participants gave 8-10 points.

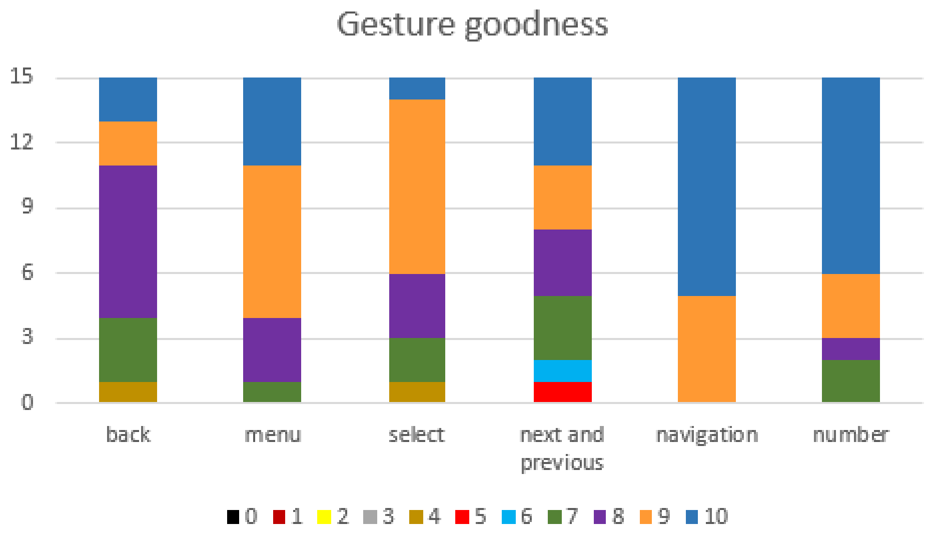


Figure 4.2: Gesture goodness

The participants used a 5-point Likert scale to answer the question if they would use the gesture recognition at home, with 1 denoting “strongly agree”, 2 “agree”, 3 “neutral”, 4 “disagree”, 5 “strongly disagree”. The average score was 3.3, the range of the scores were between 2 and 5. When the participants were bad in using gesture interaction or when the gestures couldn’t be matched with what they were looking for, they gave a lower score. This could be reduced if the user had had more learning time and would have used this system frequently. The participants who had already some experience with gestures performed better, and also gave a higher score.

4.3.3 System Evaluation

Single Ease Question (SEQ)

Nine participants gave 6 points on the SEQ (Overall, how difficult or easy was the task to complete?) and the other six participants gave 5 points. The average SEQ score is then $5,6$. This is above the standard average score, which is around five [41]. There also is a correlation between user response on the SEQ and task-time and task-completion [48], the users who gave a 5 on the SEQ, needed more time and didn't always get the right results.

System Usability Scale (SUS)

The average score of the SUS surveys are 71.25 on 100 and the standard deviation is 6.453. The score is not a percentage, the average SUS score from all 500 studies is 68 [40]. Thus the average score of this system is above the standard average. The best way to interpret this score is to convert it to a percentile rank through normalizing. This process of normalizing can be found on Figure 4.3 we can see that this system has a rank of B- and a percentile rank of around 60 percent. We have to do this because a SUS rank is not a percentile, and the average score is 68. Thus a score closer to 68, will be closer to 50 percent.

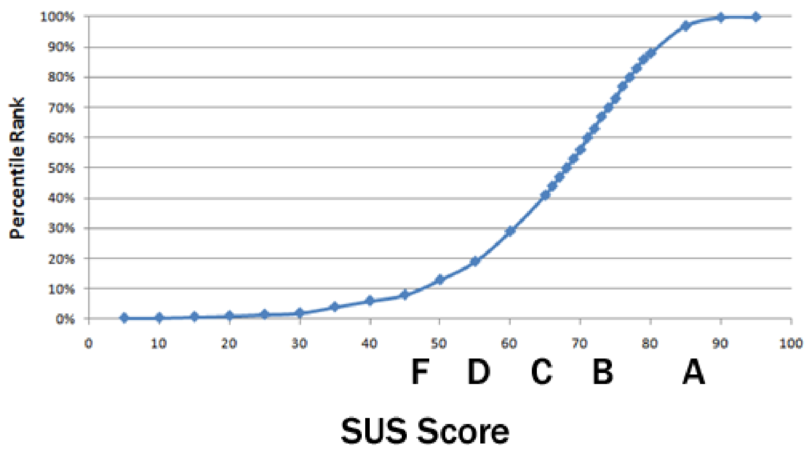


Figure 4.3: SUS curve [40]

4.4 Conclusion

We can conclude that users are open to other interaction methods than the remote control when it works with a low error rate. Most of the participants already had some bad experiences with gesture and voice interaction methods before. The participants expected more of the gesture based interaction than the voice interaction, but afterwards they were impressed of the easiness of the voice-search function. They were impressed of how easy to use the voice-search was. And how it could almost always understand their voice-search string and give the right results.

Chapter 5

Discussion & Conclusion

In this chapter, we'll discuss the results of the user tests and suggest improvements. Further, we'll compare it with previous work, present future possibilities, and a final conclusion.

5.1 Discussion of user tests results

We can conclude that suggestions for voice-search have a great impact on the search queries voiced by the participants, and should be chosen clearly without restricting users only to them. The users are tend to extrapolate examples or just use keywords which is not bad, it's better than users who are parroting the examples.
Because the error-rate was very low, the users were impressed with the voice-recognition technology. We can conclude based on the results that users are positive about voice-search if it works properly, and can see the advantages of this interaction technique.

Participants expect more of gesture recognition than voice recognition. From the results we can see that the chosen gestures are good and intuitive according to the participants. Although the average score if participants would use it at home was 3.3, this was because the gesture recognition failed more than voice recognition. The participants needed more training time to perform the gestures better. Participants are used to speaking and had more experiences with voice recognition than gesture recognition. Thus, it's important to make the participants used to gesture recognition before the experiment.

5.2 Comparison to previous work

5.2.1 Comparison to industrial systems

The voice-search of Apple TV [15] and Android TV [56] is almost similar to the library, SemanticEdge (Appendix B.1), that we've used in our implementation. The voice-search of Apple TV covers more areas: movies, music, appstore, weather, ... SemanticEdge focuses on EPG guide and VOD catalogue. Android TV also doesn't support searches in the EPG guide. SemanticEdge has some boundaries which are actually good for voice interaction with TV, SemanticEdge is only going to do a semantic analysis based on EPG and VOD data. Apple TV and Android TV both have a continuous speech feedback design, the recognized words are shown in real-time on the screen while the users speaks and are corrected while speaking. In my implementation we've used an after speech feedback design because the SemanticEdge library didn't give any feedback while the user is speaking but only returns the whole sentences once the users stops speaking.

The gesture-control in commercial systems like Samsung Smart TV (see Section 2.2.4) recognizes gestures with a camera that's on the TV. The gesture-control of Samsung is different than the gesture-control that we've implemented. The gestures of Samsung are different and uses the hand as a pointer for the TV screen. The user can "grab" a user interface element and hold it, for example to put the volume lower the user can move his hand in mid-air to the user interface element which lowers the volume, and do a long grab. While in my Leap Motion implementation for example the user doesn't point to a user interface element but just does a grab gesture with one hand and lowers his/her hand. This method of gesture-control gives the user a faster response, and gives the user a advantage using gestures. Quick access to certain UI elements like the Menu is also faster to do by writing a "M" in mid-air, then moving with the hand on the UI to the menu and grab it.

5.2.2 Comparison to previous findings

In Section 2.3, we discussed some academic literature. Stifelman et al [51] identified three challenges for speech input on TV. The first challenge was how we can convey the scope of what the system can understand. This could be done according to them by providing examples with blanks (see Section 2.3.4). In our implementation we chose to do it with examples without blanks. With blanks or without blanks the users are still stuck with the same example sentences and only change the content of it. They still extrapolate phrases (see Section 4.3.1). Another solution that we also could have used is showing random different example sentences. This way the user will see a lot of different example sentences and will not be forced to extrapolate examples.

In Section 2.3.2 it is suggested to use 2-D gestures because users prefer these gestures. They also found that users prefer either motion or hand pose gestures. Users prefer to draw the first letter of task names in mid-air to execute it. In our own findings we've used 2-D gestures because the ring device was not able to detect 3-D gestures. We've also seen that users tend to draw the first letter of a task name (e.g. "[M]enu"). Users rely on previously acquired gesture interaction methods like touch gestures (swipe right/left), this is similar to my findings where users choose to swipe left/right without the list of gestures during performing the set of gesture tasks.

In Section 2.3.5 they found that when the system recognized what the participants wanted to say, they enjoyed voice recognition. We can agree with this finding with our results, where participants gave high scores on the voice interaction part because it matched more often than they expected before the user tests.

5.3 Implications and usages

This system could be a powerful addition to any TV system. It's possible to extend the existing interaction methods with two extra methods. These methods are verified by users and their advantages are shown in user tests. It's a good usage for users with disabilities, who can use the voice recognition without touching anything. This could be done by simply implementing a "wake-up" word to trigger the voice-control. But also for everyone else, it's a great way to interact with the TV to get more of it. For example when a user doesn't know what to watch, he/she can just ask what to watch by voice. The user also can add some genres he/she likes to watch to the voice query and find what's on TV of his/her favourite genre without searching and zapping for hours. Video content could also be influenced passively. By collecting data of the voice-searches, TV manufacturers can for example put a spotlight on related video content such as TV programs, TV shows or movies.

5.4 Demonstration at IBC 2015

The voice-control part of this system is demonstrated at IBC[1] 2015 by Zappware. They demonstrated the addition of voice control to the end-to-end multiscreen platform, through a partnership with speech and natural language interpretation specialist SemanticEdge. The following comment was given by Barry Flyen in his article on VideoNet [33]: "Amsterdam has seen voice recognition demonstrations of this type before, but what was interesting about Zappware's proof-of-concept was that it was focused on solving one particular problem – searching for specific content – rather than being used as a generic navigation tool, and that the solution didn't require any 'training'.".

5.5 Future Work

The GUI of the set-top box (see Section 3.3.4) is not designed to be used for voice and gesture interaction. This could be redesigned, the suggestions/examples should be shown on the TV screen instead on the Android device. By doing this the user will focus more on the TV screen. There should be more feedback on the TV screen while speaking, e.g. Continuous speech feedback design [51] (see Section 2.3.4). The GUI could also show a tutorial of gestures at the beginning and it could be extended by showing a continuous feedback also for the gestures. This could help to improve the user's ability to do the gestures better than before.

In a living room there could be more people watching TV at the same time. When using voice interaction, this can cause some background noise. The background noise should be reduced and not recognized by the speech recognizer. A good way to extend this system and reduce the background noise a bit, is by doing the speech recognition by a smart watch which is connected to an Android application. By doing this the user will always easily hold the smart watch close to his/her mouth. Another advantage of the smart watch is that the user doesn't have to pick his/her smartphone every time he/she wants to do a voice-search, but he/she will wear the smart watch all the time and can easily do voice-searches. We've already tried this, but the smart watch was not able to connect to the internet which is needed to recognize the speech by the SemanticEdge library (see Appendix B.1.3). The device, where we have to call the speech recognizer function of the Se-

[1]International Broadcasting Convention

manticEdge library, has to be connected to the internet. But the smart watch that was available couldn't connect to the internet. There could be some workarounds like recording the voice and send it with Bluetooth to the Android device, or do the speech recognition with Google's library and the semantic analysis on the Android device with SemanticEdge. But if we do the speech recognition with Google instead of SemanticEdge (which is only based on TV and EPG data), it could increase the error-rate of recognition.

5.6 Conclusion

The user study shows that participants were impressed by the new interacting methods. There was a lot of difference before the user study and after. Six participants would not use voice recognition as interaction method with the TV but after the study they all considered to use it. Their expectation of voice recognition was low because of their bad experiences. But when a voice recognition has a very low error rate, the users will actually want to use this interaction technique more.
The gesture recognition also worked better than participants were expecting, but there was a need for more training because the time to learn was higher than the time to learn of voice-search and five participants had no experience with gesture recognition before.
With the optimizations to the GUI of the set-top box and the Android application like described in the previous section, it will be used in living rooms as an additional way of interacting with the TV.

Appendix A

Dutch Summary

A.1 Motivatie

TV kijken is een dagelijkse activiteit in ons leven geworden, terwijl alles is geëvolueerd over de jaren is de interactie met de TV stabiel gebleven. De meest dominante afstandsbediening is nog steeds de infrarood afstandsbediening. Deze methode is gelimiteerd als we het vergelijken met andere interactie methodes die we kennen van andere toestellen zoals aanraakschermen van smartphones of muis-gebaseerde computers. Gebruikers verwachten beter ontworpen input toestellen, ze verwachten informatie op de input toestel en verwachten andere interactie methodes zoals een trackbal of stemherkenning [27].

Er zijn veel televisie fabrikanten die nieuwe interactietechnieken met de TV voorzien, zoals gebaren of stemherkenning. Daarom is onderzoek naar de voorkeuren van mensen over interactietechnieken belangrijk. Meeste onderzoeken tot nu toe focussen op een specifieke actie, zoals typen met behulp van gebaren [46, 35, 37], gebaren enkel gebruiken om te navigeren [54, 34, 30], of stem interactie gebruiken voor simpele TV commando's [39].

Spraaktechnologie belooft een manier te zijn om de translatie tussen wat een gebruiker wenst en een systeem's actie te verwijderen. Gebruikers kunnen eenvoudigweg denken wat ze willen, en het gewoon zeggen. Maar het is niet mogelijk voor een spraakherkenner om de betekenis van een tekst volledig te verstaan [51]. Het gebruik van spraaktechnologie is aan het uitbreiden van mobiel naar een woonkamer omgeving waar aan TV drie meter van het gebruiker staat [38].

A.2 Doelen

Met deze thesis zoeken we voor Zappware uit hoe we nieuwe interactietechnieken zoals gebaren en stemherkenning kunnen implementeren in hun bestaande systeem. En hoe we bestaande problemen kunnen oplossen en taken kunnen vergemakkelijken met deze nieuwe interactietechnieken. Een voorbeeld hiervan is tekst invoeren, dit wordt momenteel gedaan via een on-screen toetsenbord ofwel via een afstandsbediening (SMS-style), dit kan vergemakkelijkt worden door stemherkenning waardoor de gebruiker zoekt door een zin in natuurlijke taal te vormen en dit uit te spreken. Deze interactietechnieken gecombineerd met het huidige systeem moet ook geëvalueerd worden door gebruikers met behulp van gebruikerstesten. Het is belangrijk om te weten wat de voorkeuren en gedachte zijn van gebruikers. En of ze zo een systeem daadwerkelijk thuis in hun woonkamer zouden gebruiken.

A.3 Evolutie van de afstandsbediening

In 1893 is de eerste afstandsbediening beschreven door Nikola Tesla in U.S. Patent 613809 [52]. Maar de eerste keer dat een afstandsbediening ons woonkamer betrad was de "Lazy Bone" [26]. Dit toestel was verbonden met een kabel aan de TV, het kon enkel switchen van kanalen en de TV aan en uit zetten. In 1955 kwam de eerste draadloze afstandsbediening uit, Flashmatic, het had het vorm van een geweer waarmee de gebruikers reclame's konden muten. Deze afstandsbediening was niet zo gebruiksvriendelijk. In 1956 is de Zenith Space Command afstandsbediening [26] geproduceerd door Robert Adler, deze afstandsbediening is gebaseerd op ultrasoon geluid. In 1980 werden ultrasone afstandsbedieningen vervangen door infrarood afstandsbedieningen die we nu nog steeds gebruiken. In 2001 werd de eerste draadloze afstandsbediening met een trackball ontworpen door Zenith, Z-Trak [4]. Dit afstandsbediening werkte zoals een muis, waarbij een cursor wordt weergegeven wanneer de gebruiker op de trackball klikt. In 2012 introduceerde Samsung een afstandsbediening met een touchpad en een stemherkenning functie [8]. LG introduceerde in 2013 een afstandsbediening waarmee de gebruiker kon interageren met behulp van stem, gebaren, trackball en pointing [5].
Tegenwoordig zijn er vier populaire afstandsbedieningen, de afstandsbedieningen van Apple TV [15] en Android TV [14] onderscheiden zich van de vorige afstandsbediening door niet enkel stemherkenning te doen maar ook te voorzien van natuurlijke taal herkenning en semantieke analyse. De Apple

TV afstandsbediening heeft ook nog een touchpad waarmee gebruikers kunnen navigeren in de menu door simpele swipes. De Amazon Fire TV Remote ondersteunt ook stemherkenning voor te zoeken maar dan zonder natuurlijke taal herkenning, dus er zijn vaste commandos waaraan de gebruikers zich moet houden. Dit geldt ook voor de Samsung Smart TV [12, 10], Samsung ondersteunt stemherkenning maar het wordt voor acties gebruikt. Samsung heeft ook een gebarenherkenner waarmee de gebruiker de TV kan bedienen met gebaren op afstand (zie Figuur 2.2.4 voor alle mogelijke gebaren).

A.4 Implementatie (Stage)

Voor mijn stage bij Zappware kreeg ik de taak om hun bestaande applicatie uit te breiden door twee nieuwe interactietechnieken toe te voegen, stem en gebaren interactie. Zappware wilt hiermee focussen op innovatieve interactie methoden met mobiele toestellen en TV's. "Voice-control" moet gezien worden als een aanvullende manier van interactie. Waarbij het belangrijk is om te kunnen zoeken in de EPG en VOD cataloog met behulp van stemherkenning, ook is het belangrijk om snelle toegang te hebben tot bepaalde menu's met stemherkenning. Gesture-control heeft het potentieel om de huidige afstandsbediening te vervangen. Daardoor is het belangrijk om onderzoek te doen naar de precisie van de input toestellen en ook hoe we normale gebaren kunnen onderscheiden van gebaren die bedoeld zijn om te interageren met de TV.

We hebben met Zappware besloten om de bibliotheek SemanticEdge (zie Appendix B.1) te gebruiken voor voice-control en een Logbar Ring Zero [18] voor gesture-control. De SemanticEdge bibliotheek is een Android bibliotheek daarom heb ik gebruik gemaakt van de bestaande Android applicatie van Zappware om voice-control hierop te implementeren. Om de gesture-control te implementeren met Logbar Ring Zero moest ik gebruik maken van de iPhone applicatie van Logbar omdat er nog geen Android applicatie beschikbaar was. Deze beide applicaties verzenden stem en gebaren (zoek)acties door naar de Node.js server (zie Appendix B.2). Deze server identificeert de acties en zendt het door naar de set-top box waar er een actie wordt gedaan of zoekresultaten (bij voice-search) worden getoond.

Één van de belangrijkste taken bij voice-control was om een zoekfunctie te implementeren waarmee gebruikers konden zoeken met behulp van natuurlijke taal. Hiervoor heb ik gebruik gemaakt van SemanticEdge Dialog client API (zie Appendix B.1.3) dit is een cloud gebaseerde stemherkenner en doet ook de semantieke analyse van de herkende tekst voor Android. De GUI moest ook aangepast worden aan de nieuwe interactie techniek, we hebben een extra knop toegevoegd voor voice-search en een pop-up. Dit pop-up bevat een retry knop, een animatie voor de amplitude van de stem en een lijst van suggesties/voorbeelden.

Voordat ik de Logbar Ring Zero [18] heb gebruikt bij de implementatie van gesture-control, heb ik eerst gebruik gemaakt van een Leap Motion omdat de ring nog niet beschikbaar was. Leap Motion kan 3-D gestures detecteren wat meer gestures oplevert. Omdat het ook mogelijk is om direct feedback te krijgen van een Leap Motion heb ik een gesture geímplementeerd voor volume waarbij de gebruiker zelf bepaalt met zijn hand hoeveel volume er toegevoegd of afgetrokken moet worden (zie Figuur 3.5). Om van kanaal te switchen heb ik gebaren gebruikt waarbij de gebruiker met zijn/haar hand naar links of naar rechts moet bewegen. En om te switchen naar een specifiek kanaal, het nummer van de kanaal te schrijven in de lucht.
Met een Logbar Ring Zero is het enkel mogelijk om 2-D gestures te detecteren met feedback nadat een gesture is uitgevoerd door de gebruiker. In de volgende tabellen 3.1-3.2 zijn de gestures beschreven. Sommige van deze gestures zijn gebaseerd op een elicitatie studie op Leap Gesture [51] (meer over deze studie kan gelezen worden in sectie 2.3.2).

A.5 Gebruikerstesten

Voordat elke gebruiker begon aan de gebruikerstest heb ik hen gevraagd om een vragenlijst (zie Appendix C.1) in te vullen over hun achtergrond en over hun ervaringen met de twee interactietechnieken stem en gebaren. Later hebben ze meegedaan aan gebruikerstesten die we eigenlijk in twee kunnen opdelen, stem en gebaren. Er hebben 15 personen meegedaan aan deze gebruikerstesten, bij elke gebruikerstest hadden de gebruikers een lijst van taken voor zich (taken zijn te vinden in Appendix C.2). De doel van deze gebruikerstesten was om de voorkeuren van de gebruikers te verstaan voor interageren met de TV. Ook was het belangrijk om te begrijpen hoe de ze zich voelen bij het gebruiken van interactietechnieken alleen en in het bijzijn van andere. Het valideren van de gebruikte gestures op goedheid en intuïtiviteit was belangrijk, en ook het valideren van de stemherkenning. Ik was geïnteresseerd of de gebruikers van gedachte zouden veranderen over de twee interactietechnieken en over het hele systeem voor dat ze begonnen aan de gebruikerstesten en na de gebruikerstesten. Of ze zo een systeem thuis zouden gebruiken, in hun dagelijkse leven.

Een 40-inch Philips Smart TV was aangesloten met een laptop dat op Microsoft Windows 8.1 draait. De Logbar Ring Zero was geconnecteerd via bluetooth met een iPhone om de gebaren te herkennen en het door te sturen naar een Nodejs server dat runt op dit laptop. Een Android toestel werd gebruikt voor stemherkenning, de resultaten werden weer doorgestuurd naar de Nodejs server. Deze server verzond het vervolgens naar een HTML5 TV UI dat aan het runnen was op de laptop en getoond op de TV.

Na de voice-control experiment kan ik concluderen dat de suggesties voor voice-search een grote impact hebben op de zinnen die de gebruikers vormen om te zoeken. Gebruikers hebben 47,62% van de zinnen geëxtrapoleerd van de suggesties en 8,57% van de zinnen waren hetzelfde als de suggesties. 26,67% waren keywords, volgens één gebruiker gebruikte hij dit omdat hij te lui was om volledige zinnen te vormen. In het algemeen waren de meeste deelnemers onder de indruk hoe goed het overeenkwam met wat ze wouden zoeken, dit komt omdat ze eerder slechte ervaringen hadden met stemherkenning technologieën. Na de experiment veranderde veel deelnemers van gedachte, en zouden zo een stemherkenning technologie thuis gebruiken.

In het algemeen gaven de deelnemers hoge punten op de gesture goodness vragenlijst, en ook op hoe intuïtief de gestures zijn. Er was enkel meer learning time nodig voor gebruikers die minder kennis hadden van gestures, omdat zij moeilijkheden hadden om sommige gestures uit te voeren.

A.6 Conclusie

De gebruikerstesten tonen dat deelnemers onder de indruk waren van de nieuwe interactie methoden. Er was een groot verschil voor en na de gebruikerstesten. Zes deelnemers zouden een soortgelijke stemherkenning technologie met de TV niet gebruiken voor de experiment, maar na de gebruikerstesten zouden ze het allemaal overwegen om te gebruiken. De verwachtingen van de deelnemers over stemherkenning was vrij laag omdat de gebruikers slechte ervaringen hadden. Maar wanneer het een lage error rate heeft willen de gebruikers deze interactie methode gebruiken.
De gesture recognition werkte beter dan dat de gebruikers verwachtten, maar er was een nood aan praktijk omdat de leertijd hoger was dan stemherken-

ning en vijf deelnemers hadden nog geen ervaring met gesture recognition. Met de nodige optimalisatie aan de grafische user interface van de set-top box en de Android applicatie, zou deze twee interactietechnieken gebruikt worden in woonkamers als een aanvullend interactiemethode met de TV.

Appendix B

Libraries

B.1 SemanticEdge

The SemanticEdge Natural-Language Voice User Interface (NL-UI) offers a complete solution for integrating a multi-modal user-interface into existing TV and Video platforms. It consists of two components: the Interpretation and Dialog Platform and the SE Dialog Client. The SE Dialog Client should be integrated into the remote control app to use the functionality of the NL-UI. The functionalities of SemanticEdge NL-UI for Smart TV are the

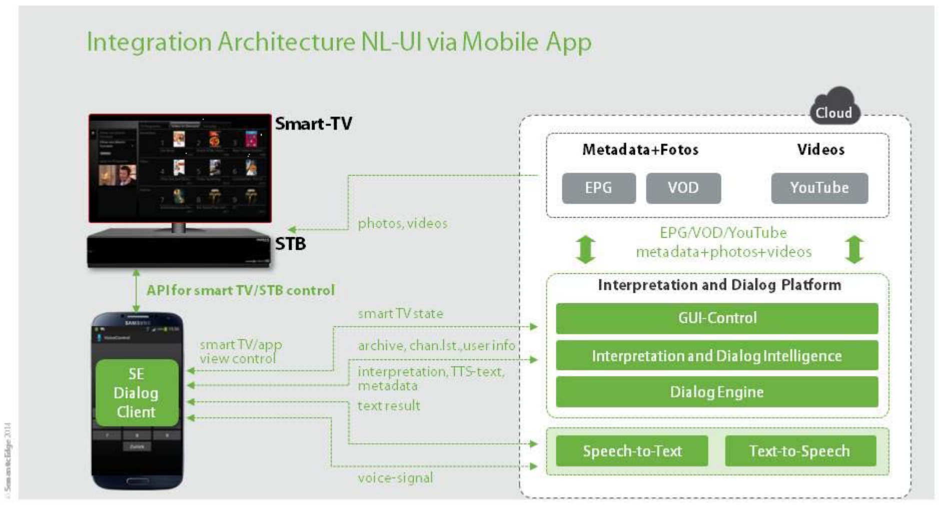

Integration Architecture NL-UI via Mobile App
Smart-TV
STB
Cloud
Metadata+Fotos
Videos
EPG
VOD
YouTube
photos, videos
EPG/VOD/YouTube metadata+photos+videos
API for smart TV/STB control
Interpretation and Dialog Platform
GUI-Control
Interpretation and Dialog Intelligence
Dialog Engine
smart TV state
smart TV/app view control
archive, chan.lst., user info
interpretation, TTS-text, metadata
text result
SE Dialog Client
Speech-to-Text
Text-to-Speech
voice-signal
© SemanticEdge 2014

Figure B.1: Integration architecture NL-UI via mobile app [19]

following:

1. It provides speech recognition and natural-language interpretation functionality covering the key domains and use cases in smart TV remote control and media search.

2. It can provide expressive control functionality for the TV monitor in order to adapt the GUI to the natural-language commands and search results.

3. In order to provide these two functionalities the SemanticEdge NL-UI for Smart TV integrates the metadata sources for EPG, VOD and YouTube as well as relevant other user and device information that may affect the remote control or search functionality and the dialog. This includes the complex processes of regular metadata-updates and their processing for the presentation.

B.1.1 Specifics of a Natural-Language User Interface

The natural-language user interface has to cover a wide functionality, it has to cover all possible use-cases. Because users need to be free in their voice query, it should be possible to ask open-ended questions. You can't restrict users in their vocabulary and ask them to learn certain commands before using it. But still the user wants to have some assistance in what they could be asking. The user can do the following commands:

- A remote control command like switching the channel, changing the volume, play/pause a movie,...
- Search for EPG, VOD, Youtube content.
- Navigate through the displayed content.
- Ask for help.

SemanticEdge needs access to some metadata (EPG,VOD,Youtube) to do semantic interpretation of a voice query like for example "show me the program on BBC1 tonight". SemanticEdge has also information of actors and directors in order to search and also give a negative feedback when the known actors are not in the metadata.

B.1.2 Covered domain and use cases

SemanticEdge provides speech recognition and semantic interpretation in the following domains: Remote Control, EPG-Search, Video-on-Demand search and YouTube search. Within these domains the following use cases are supported where natural-language commands can be interpreted.
In the EPG the user can search for: title or substring of title, date, time, channel, actor, director, role, genre/keyword and country.
Some examples:

- "Show me the program of Discovery Channel tomorrow at 8 p.m."
- "Any hospital dramas on air right now"
- "When are Channel 4 News showing tonight?"
- "Are there any movies with Tom Cruise showing in the next days?"

The users can search in Video-on-Demand for: title(+substring of title), actor, director, role, genre/keyword, country, release date/time period.
Some examples:

- "Show me all movies by Quentin Tarantino"
- "I am looking for western movies from the sixties"
- "Harry Potter and the Deathly Hallows part II"
- "Show me all movies with Robin Williams"
- "French movies of the 80s"

The use cases for Remote Control are: power on/off, switch channel, volume control, start/stop/forward/rewind player, record program.

B.1.3 Dialog Client Library API

The Dialog Client Library API of SemanticEdge is a library only for Android. It establishes a bidirectional communication path for a coupling of the device and the cloud-based speech recognition and semantic interpretation. We'll explain some important functions of the library first:

- public static void **connectThisTV**(Context context, Handler messageHandler, String tvID, String smartID, String expirationTimer, Locale locale, String userFirstName, String userLastName, String timeZone, String timestampClient)

This function should always be used first, it initializes the SE Dialog connection and registers the TV or STB. (connects to the cloud speech-recognizer)

- public static void **startVoiceRecognition**(String timestampClient)
  This method will start the voice recognition and returns a speech-to-text result with a confidence value.
- public static void **doSemanticAnalysis**(String handle, String speech-text, String currentState, String timestampClient)
  With this method we'll do the semantic analysis of a recognized text within a given dialog state. This returns the results in JSON format.

The last function returns three important values: command_type, tv_command and monitor_type. The command_type is either a "TVCommand" or a "Search". If the command_type is of the type "TVCommand", the tv_command value will be of this structure:

```
tv_command: {
    volume : "70"
}
```

In this example the attribute is volume, but it could also be channel, power, mute, record, memorize, player (pauze, continue, forward, rewind, start_over, exit), player(VOD) or player(YouTube).
If the command_type is of the type "Search", the monitor_content is returned which contains some attributes. One of the attribute is the "view" object which contains the main attributes for presenting the result data on the screen. The "context" attribute shows the navigation search path, the "prompt" attribute contains a reply/confirmation generated by the library that could be prompted to the user as a text or via text to speech. The last attribute is the "data" object, it contains the results as a list of metadata objects. Depending on the request the object may belong to different domains like VOD, EPG or YouTube.
An example of the monitor_content structure is as follows:

```
monitor_content: {
    view: {
        name : "OVERVIEW"
        data_type: "EPG"
        groups: ["12:00 - 14:00",
        "14:00 - 15:00",
        "15:00 - 17:00"],
        grid: {
            columns: 3,
            lines: 3
        },
        page_count: {
            total: 5,
            current: 2
        }
    },
    content: "TV Program/BBC",
    prompt: "Program on BBC.",
    tts: "Program on BBC.",
    data: [{},{},...]
}
```

The data list could contain EPG items which has the attributes: program_id, language, title, duration, episode_number, episode_title, season_number, release_year, genres, image_url, start_time, end_time, provider, group, domain and reference_number. It could also contain a VOD item which has almost the same attributes and it could also contain YouTube items.

B.2 Node.js

Node.js [17] is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. It uses an event-driven, non-blocking I/O model which makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
With the Node.js there is no Apache to listen for incoming connections and return HTTP status codes. The user needs to handle this by himself, but the result is a high performance web application.
The advantages of Node is firstly his performance and scalability. Node is really fast, that's an important requirement for most of the developers. Node is also perfect for offering a RESTful API.
The downside of Node is that it's not easy to deploy Node on existing hosts. If you have a shared web hosting, you can't simply upload a Node app and expect it to work. VPS and dedicated servers are better positioned, you can install Node on them.

B.3 Socket.IO

Socket.IO [21] is a JavaScript library for real-time web applications. The communication is bidirectional and real-time between web clients and a server. It consists of two parts, one part is the client-side library that runs in the browser, and a server-side library for Node.js.
It uses the WebSocket protocol with polling as a fall back option, while providing the same interface.

One of the advantages of using Socket.IO is that the connection is transparent. And it will automatically upgrade to WebSocket if possible. So the user only needs knowledge of Socket.IO.
A disadvantage of using Socket.IO is that there are no fall back options like other real-time protocols. It's also not possible to connect a WebSocket client with a Socket.IO server. Both client and server side should be using Socket.IO in order to connect and talk with each other.

There is also an Android library of this library named "Socket.IO v1.x Client Library for Java" which we've used in my implementation for the Android application client side.

Appendix C

Surveys

C.1 First survey (before the experiment)

The survey is shown on the next pages.

<u>Use of Voice Control/Search applications and gestures</u>

This is a survey to check what users think about voice and gesture interaction, and to know how often they use it.

1. Age?

- ☐ < 14
- ☐ 14-18
- ☐ 19-25
- ☐ 26-40
- ☐ > 40

2. Gender?

- ☐ Male
- ☐ Female

3. Student or working?

- ☐ Student, faculty: …………………………
- ☐ Work, sector: ………………………………
- ☐ None of the above

4. Do you have a smartphone or a tablet?

- ☐ Yes
- ☐ No (go to question 7)

5. Do you use the voice interaction app on your smartphone/tablet (Ex. Siri, OK Google Now)

- ☐ Yes
- ☐ No

6. For which purpose are you using the voice interaction apps?

6.1 To call someone (ex. Call Justin, Call home, Call my wife, etc)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.2 To ask for directions (ex. "Directions to home", "Show me directions to Philadelphia Pennsylvania", etc)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.3 To play music (ex. "Play Jazz music", "Pause music, etc)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.4 To ask the weather (ex. “What’s the weather like today?”)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.5 To search on web (ex. “Search for baked chicken recipes”)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.6 To send a text message (SMS) (ex. “Send a message to Justin”

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.7 To read text messages (SMS) (ex. “Read my new messages”

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.8 To set an alarm/timer (ex. “Set an alarm for 7 AM”

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.9 To change phone settings (ex. “Turn on Wi-Fi”, “Turn on airplane mode”)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.10 To open an application (ex. “Launch Spotify”)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.11 To create notes (ex. “Create note: Door pass code is one two three two”)

○ daily ○ weekly ○ monthly ○ yearly ○ never

6.12 To search contacts (ex. “Show Alicia ‘s home email address”)

○ daily ○ weekly ○ monthly ○ yearly ○ never

7. Do you use voice interaction in another environment? (ex. some cars have voice interaction functionalities, gps to ask directions, at home, controlling lights with voice, to control your TV with voice, ...)

- ☐ Yes, which? …………………
- ☐ No

8. **Would you use voice interaction (natural language understanding) to search on your TV content? (Ex. “Show me the latest episode of Breaking Bad”, “Is there any comedy movies tonight?”, “Switch to BBC News”, “Show me all movies with Jim Carrey”, “Football on TV”)**

   - ☐ Yes
   - ☐ No

9. **Do you use voice interaction in presence of others?**

   - ☐ Yes
   - ☐ No

10. **How do you feel when you use voice interaction?**

   ………
   ………
   ………
   ………
   ………
   ………
   ………

11. **How do you see the future of voice interaction? And what do you expect of the voice interaction in the future?**

   ………
   ………
   ………
   ………
   ………
   ………

12. **Have you used gesture based interaction before?**

   - ☐ Yes, ……………………
   - ☐ No

13. **How do you feel when you use gesture based interaction?**

   ………
   ………
   ………
   ………
   ………

14. **What do you think of the future of gesture based interaction?**

   ………
   ………
   ………
   ………
   ………

C.2 Tasks

The tasks are shown on the next page.

<u>Tasks during the test</u>

<u>1. Voice</u>

1. Search for
    - movies of the genre comedy
    - movies with a specific actor
    - a specific movie
    - football programs on TV
    - programs about cooking this week
    - programs on a specific channel
    - only American movies
    - “choose your own search sentence”

2. Switch to Channel: BBC World News
3. Put the volume higher

<u>2. Gesture</u>

1. Navigate to The menu
2. Navigate to the list of channels
3. Navigate to channel nr. 3 in the list
4. Navigate to channel 8 by drawing the nr
5. Navigate to a channel of your own choice (by drawing)
6. Switch between channels with next and previous gestures
5. Change the interface language to French

C.3 Voice recognition survey

Voice

1. **How did you find the search strings?**

1.1 movies of the genre comedy

○	○	○	○	○
very easy	easy	neutral	hard	very hard

1.2 movies with a specific actor

○	○	○	○	○
very easy	easy	neutral	hard	very hard

1.3 a specific movie

○	○	○	○	○
very easy	easy	neutral	hard	very hard

1.4 football programs on TV

○	○	○	○	○
very easy	easy	neutral	hard	very hard

1.5 programs about cooking this week

○	○	○	○	○
very easy	easy	neutral	hard	very hard

1.6 programs on a specific channel

○	○	○	○	○
very easy	easy	neutral	hard	very hard

1.7 only American movies

○	○	○	○	○
very easy	easy	neutral	hard	very hard

2. **Did your search strings matched what you were looking for?**

○	○	○	○	○
never	sometimes	neutral	often	always

3. **Did you expected the search string would give the results?**

○	○	○	○	○
never	sometimes	neutral	often	always

4. **Would you use the voice recognition at home?**

○	○	○	○	○
never	sometimes	neutral	often	always

5. **Comments**

……
……
……
……
……
……
……

C.4 Gesture survey

The survey is shown on the next pages.

<u>Gestures</u>

1. **How did you expected the navigation gestures would work?**

○	○	○	○	○
very good	good	neutral	bad	very bad

2. **How did the navigation gestures actually work?**

○	○	○	○	○
very good	good	neutral	bad	very bad

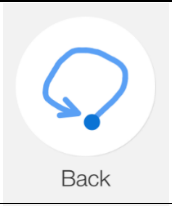
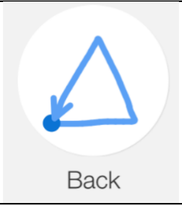
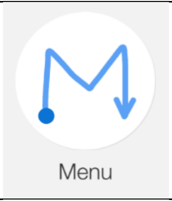
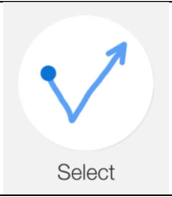
3. **How intuitive are the gestures?**

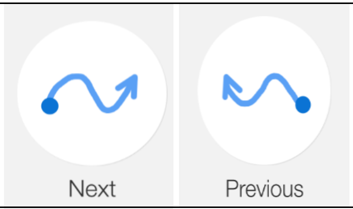
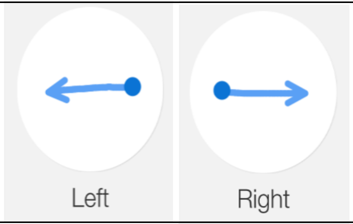
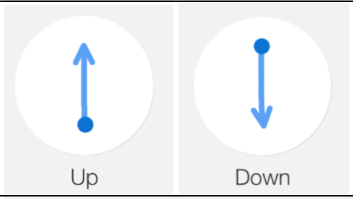
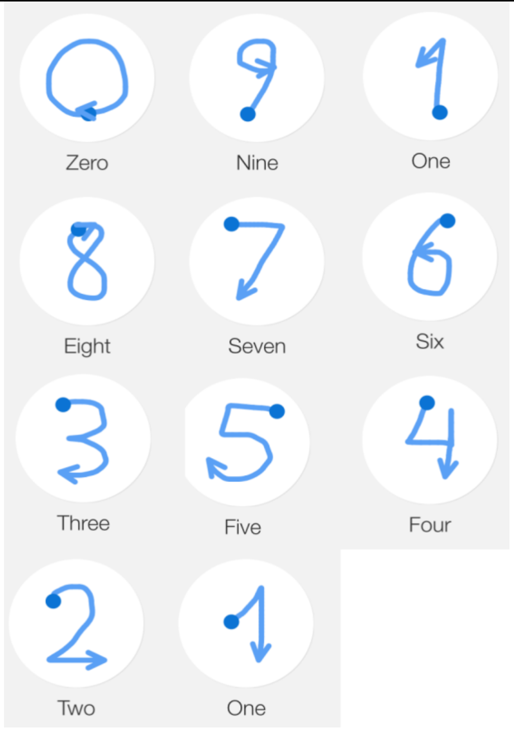
○	○	○	○	○
very good	good	neutral	bad	very bad

4. **Would you use the gesture recognition at home?**

○	○	○	○	○
strongly agree	agree	neutral	disagree	strongly disagree

5. **Give a score for each gesture and would you change it?**

Gesture	**Score /10**	**I would change this gesture into this… (optional)**
 		
		
		

 Next Previous		
 Left Right		
 Up Down		
 Zero Nine One Eight Seven Six Three Five Four Two One		

6. Comments

C.5 System Evaluation

The survey is shown on the next pages.

System

SEQ: Overall, how difficult or easy was the task to complete?

1	2	3	4	5	6	7
○	○	○	○	○	○	○
very difficult						very easy

1. **I think that I would like to use this system frequently.**

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

2. **I found the system unnecessarily complex.**

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

3. **I thought the system was easy to use.**

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

4. **I think that I would need the support of a technical person to be able to use this system.**

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

5. **I found the various functions in this system were well integrated.**

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

6. I thought there was too much inconsistency in this system.

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

7. I would imagine that most people would learn to use this system very quickly.

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

8. I found the system very cumbersome to use.

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

9. I felt very confident using the system.

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

10. I needed to learn a lot of things before I could get going with this system.

1	2	3	4	5
○	○	○	○	○
strongly disagree				strongly agree

Appendix D

Natural Language Understanding

D.1 Definition

Natural language understanding (NLU) is subfield of Natural language processing (NLP) that is about the comprehension of machine reading. The goal of this subfield is actually to understand or "interpret" an input text (language). The Natural Language Understanding process is basically to translate the input text to an unambiguous formal language representation of the input text.

Natural language understanding is as old as computers itself, but recently NLU interfaces are in demand and with the recent advances in machine learning, it's a hot topic again.

The humans are the best example of natural language understanding; when we read a sentence on this page for example the words we read are absorbed into our brains and transformed into concepts, this enters into a rich network of previously-acquired concepts (knowledge). We are actually doing natural language understanding all the time.

D.2 Early History

In early 1930s Alan Turing defined the universality of computation, which states that any computation could be done on a Turing machine. So should it be possible for a computer to understand language?

In 1950 Alan Turing defined the Turing test, which can evaluate if a computer actually understands language.
The computer was becoming more involved in the study of language, in 1949 Warren Weaver [25] proposed to solve the world-wide translation problems with the computers. The algorithm was as follows: every word was looked up in a bilingual dictionary, an equivalent word is chosen in the output language, after processing each sentence, the strings of words are arranged to be right in the output language's word order.
This was not the most effective/optimal way, because there were many problems in selecting equivalent words and arranging the sentence. After some years the concept of machine translation changed to "understanding", the machine has to understand the meaning of the sentence.
The first most known natural language understanding program is the STUDENT that is developed by Bobrow (1964) [28]. This program could solve algebra problems of high school level. An example query of this program could be something like: "If the number of customers Tom gets is twice the square of 30% of the number of advertisements he runs, and the number of advertisements is 55, then what is the number of customers Tom gets?". The STUDENT program is just going to process the query by doing simple pattern matching (section D.3.1) to translate it into an equation-solving program.

D.3 NLU Methods

D.3.1 Pattern Matching

Pattern matching method [36] interprets the input as a whole, and doesn't build the interpretation of it by combining structure and the meaning of the words. This is a holistic approach, by matching patterns of words with the input we can obtain the interpretation of the input. There is an interpretation associated with each pattern, the derived interpretation is the attached one to the pattern that matched it. One example of this method is the ELIZA system, this is the earliest system of this method. This system shows how easy it is to mimic intelligent behaviour. ELIZA doesn't understand actually what is said but can give convincing replies to the user. This system doesn't construct an internal representation of the input but it goes directly from the input to the reply, which is like the interpretation of the input. The input is matched with a small set of patterns, each associated with some replies. The task of ELIZA was to simulate a Rogerian psychologist interviewing a patient. An example of this kind of pattern with two

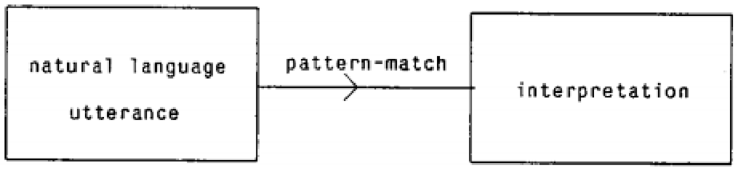


Figure D.1: Parsing by pattern matching [36]

variables X and Y:

X always Y	Can you think of a specific example?
	When?
	Really, always?
X you Y me	Why do you think I Y you?
	You like to think I Y you, don't you.
	Suppose that I did Y you. What would that mean?

An example of a conversation between patient(P) and ELIZA(E) with this pattern:

P: People always stare at me	(matches first pattern)
E: Can you think of a specific example?	
P: For instance you keep staring at me	(matches second pattern) with Y matching "keep staring at"
E: Why do you think I keep staring at you	(first reply for second pattern with appropriate substitution for Y)
P: Because you hate me	(second pattern again with Y matching "hate")
E: You like to think I hate you, don't you.	(next (second) reply for second pattern)

D.3.2 Syntactical Driven Parsing

This is actually the opposite of pattern matching, with syntactical driven parsing we have to split the sentence. The most known form of this method is the parse tree. Here is an example of this form for the sentence "the rabbit nibbled the carrot". The tree shows that the sentence consists of a noun phrase(NP), and a verb phrase(VP).

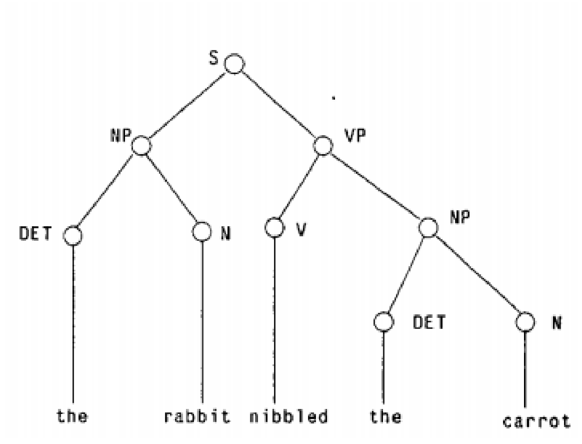


Figure D.2: Parse tree for “the rabbit nibbled the carrot” [36]

D.3.3 Grammar parsing

Grammar is a scheme for defining the sentences how it is allowed in a language [25], this indicates also the syntactic rules of combining words into well-formed phrases and clauses. In 1957 Chomsky introduced the theory of generative grammar, this had an influence on the linguistic research. In natural language understanding grammar is used to decompose the sentence (parsing) and put it like that in the input to help to determine the meaning of the sentence. There are several types of grammar used in NLU: phrase structure grammars, semantic grammars, transformational grammars and case grammars. Parsing means the delinearization of linguistic input by using grammatical rules and other sources of knowledge. This is used to get the functions of the words in the sentence to create a data structure. It's really complex to design a parser, not only at the implementation side but also in theory.

D.3.4 Rule-Based Systems

The rule-based systems [43] are the first natural language understanding systems. These are emerged in the early 1960s. For example at MIT, a PhD student built a system for his thesis to solve algebra word problems of high-school algebra books. Another example is the LUNAR system, which is developed by Bill Woods in the early 1970s. This system provided a natural-language interface into a database about moon rocks that had been brought back on the recent Apollo 11 mission. So the users (scientists) could

ask the systems queries like “list all the rocks that contain chromite”. So it understands the sentence and transforms it into a SQL query.
Another example is the system SHRDLU, this system allowed users to interact in a block environment. It could also answer questions and execute actions. The user could move blocks for example, “Find a block that is taller than the one you are holding and put it into the box.” In figure D.3 this query should do the following action: It has to move the small green block (that is on the top of the blue block) away, take the blue block and put it in the box.

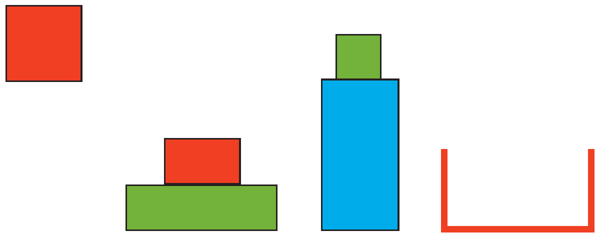


Figure D.3: SHRDLU system: blocks interface [43]

These examples were a great achievement in their time, they could handle complex linguistic phenomena and integrate syntax and semantics.

D.4 Complexity

Why is it actually hard for a computer to understand natural language understanding? Well if we compare a programming language with a natural language we can see too many differences. For example if we compare the programming language Python with the natural language English. Python is unambiguous, lambda x : x.split(”,”)[0:4] this expression in python has exactly one meaning. But English can be ambiguous and really vague. We also use a lot of references in our sentences, so we can’t understand the sentence just by reading each word. We perform this completion based on knowledge about the world that we cultivate throughout our lives. Computers didn’t have this information, and it becomes really difficult for them to understand sentences where the words alone don’t mean so much.
There are also similarities between English and Python, compositionality for example. The definition is that the meaning of the whole is derived from the meaning of the parts. In Python for example if we compute “(9 + 1) - 3”, the interpreter is going to compute this by understanding numbers,

operators, and combination rules. And humans understands for example "blue pen" by understanding the constituent words.

D.5 Statistical techniques

Until the early 1990s the natural language process research had been rule-based. After the increasing of data availability and computing power, natural language processing went statistical. With this new system we have to think differently than rule-based. There are some requirements, we firstly should collect examples of desired input-output behaviour of the program. Secondly we have to write a partial program with unknown parameters, the system should use a machine learning algorithm to automatically tune these parameters based on the examples.
An example of this system, in the artificial intelligence, was the checkers program of Arthur Samuel (1959). This program learned to play from its own moves and game outcomes.
Machine learning plays an important role today for speech recognition, but also in robotics, medical diagnosis, ad placement, etc. It also drives many natural language process tasks:

- Part-of-speech tagging (e.g. identifying "Brussels" as a proper noun)
- Named-entity recognition (e.g. identifying "Brussels" as a location)
- Syntactic parsing (e.g. identifying "Brussels" as the direct object of a sentence)
- Machine translation (e.g. converting "Brussels" to "Brussel" in Dutch)

Question answering is a task which is close to language understanding (e.g. "What is the largest city in Belgium?" the system responds with "Brussels"). A question answering system uses the question to retrieve relevant web pages using a search engine, extracts candidate answers from those pages, and then scores and ranks the answers.

D.6 Conclusion

The rule-based system and statistical views are complementary. The best combination is the logically sophisticated representations of rule-based systems with the robustness and automation of statistical techniques.

It's better to use a logical form for e.g. the sentence "largest city in Belgium", argmax:City(x)∧LocatedIn(x,Belgium)∧Population(x) instead of treating the words purely. This mapping is learned from data and it's not manually coded.
In the last years, they found some new techniques. Like new machine learning techniques whose input data consists of answers to questions rather than logical forms. By using crowdsourcing there is a much larger question-answering datasets.

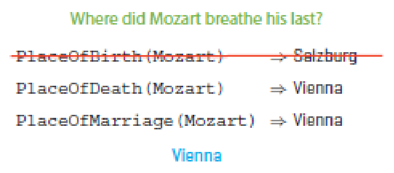

Where did Mozart breathe his last?
~~PlaceOfBirth(Mozart) ⇒ Salzburg~~
PlaceOfDeath(Mozart) ⇒ Vienna
PlaceOfMarriage(Mozart) ⇒ Vienna
Vienna

Figure D.4: Question-answer pairs example [43]

Figure D.4 illustrates a simplified version of how a system would conceivably learn from two question-answer pairs. For each question, the system generates a set of possible logical forms and executes each one to yield an answer. The logical forms that do not yield the correct answer are discarded. Based on the remaining logical forms, the system would find that "breathe his last" occurs more consistently with "PlaceOfDeath" than the alternatives and choose this mapping.

Bibliography

[1] https://parakim22.files.wordpress.com/2015/06/unnamed-file1.jpg.

[2] http://www.pushclicktouch.com/blog/wp-content/uploads/2007/03/zenith600_top.jpg.

[3] http://i10.ebayimg.com/04/i/001/32/90/32a9_12.JPG.

[4] Fascinating facts about the invention of wireless tv remote control by robert adler in 1956. http://www.ideafinder.com/history/inventions/remotectl.htm, 2006.

[5] The evolution of a remote control. http://techtalk.currys.co.uk/tv-entertainment/the-evolution-of-a-remote-control/, 2013.

[6] All of the smart tv gestures. http://www.samsung.com/ph/smarttv/common/guide_book_3p_si/main.html, 2014.

[7] Amazon fire tv. http://www.amazon.com/Fire-TV-streaming-media-player/dp/B00CX5P8FC, 2014.

[8] Evolution of samsung tv remote controls. http://global.samsungtomorrow.com/evolution-of-samsung-tv-remote-controls/, 2014.

[9] Review: Amazon fire tv. http://fortune.com/2014/05/07/review-amazon-fire-tv/, 2014.

[10] Samsung smart tv voice control. http://www.samsung.com/ph/smarttv/voice_control.html, 2014.

[11] Use your voice to search amazon fire tv devices. http://www.amazon.com/gp/help/customer/display.html?nodeId=201497650, 2014.

[12] Voice control. http://www.samsungdforum.com/UxGuide/2014/03_input_method.html, 2014.

[13] Voice search for google tv. https://support.google.com/googletv/answer/2853568?hl=en&ref_topic=2880324, 2014.

[14] Android tv. https://www.android.com/tv/, 2015.

[15] Apple tv. https://www.apple.com/tv/experience/, 2015.

[16] Express for node.js. http://expressjs.com/, 2015.

[17] Node.js. http://nodejs.org, 2015.

[18] Ring zero by logbar. http://logbar.jp/ring/en, 2015.

[19] Semanticedge nl-ui for smart tv and smart home, 2015.

[20] Six decades of channel surfing: History of the tv remote control. http://www.zenith.com/remote-background/, 2015.

[21] Socket.io. http://socket.io/, 2015.

[22] Sony bravia - how to control your tv with the touchpad remote control for sony's android tv. http://www.youtube.com/watch?v=cdFUyR8FXWM, 2015.

[23] Use siri on your apple tv. https://support.apple.com/en-us/HT205300, 2015.

[24] Coordinate systems. https://developer.leapmotion.com/documentation/java/devguide/Leap_Coordinate_Mapping.html, 2016.

[25] Avron Barr. Natural language understanding. *AI Magazine*, 1(1):5, 1980.

[26] Mary Bellis. History of the television remote control. http://inventors.about.com/od/rstartinventions/a/remote_control.htm.

[27] Regina Bernhaupt, Marianna Obrist, Astrid Weiss, Elke Beck, and Manfred Tscheligi. Trends in the living room and beyond: Results from ethnographic studies using creative and playful probing. *Comput. Entertain.*, 6(1):5:1–5:23, May 2008.

[28] Daniel G Bobrow. Natural language input for a computer problem solving system. 1964.

[29] Nathan Chandler. History of the remote control: The downfall of western civilization. `http://www.123HelpMe.com/view.asp?id=69756`, 2015.

[30] Ming-yu Chen, Lily Mummert, Padmanabhan Pillai, Alexander Hauptmann, and Rahul Sukthankar. Controlling your tv with gestures. In *Proceedings of the international conference on Multimedia information retrieval*, pages 405–408. ACM, 2010.

[31] Niloofar Dezfuli, Mohammadreza Khalilbeigi, Jochen Huber, Florian Müller, and Max Mühlhäuser. Palmrc: imaginary palm-based remote control for eyes-free television interaction. In *Proceedings of the 10th European conference on Interactive tv and video*, pages 27–34. ACM, 2012.

[32] Daniel Engber. Why are remote control so terrible. `http://www.slate.com/articles/life/design/2012/06/the_history_of_the_remote_control_why_are_they_so_awful_.html`, 2012.

[33] Barry Flynn. Zappware demonstrates voice recognition to support multiscreen content discovery. `http://www.v-net.tv/zappware-demonstrates-voice-recognition-to-support-multiscreen-content-discovery`, 2015.

[34] William T Freeman and Craig Weissman. Television control by hand gestures. In *Proc. of Intl. Workshop on Automatic Face and Gesture Recognition*, pages 179–183, 1995.

[35] Nathan Godard, Isabelle Pecci, and Poika Isokoski. Weslide: Gestural text entry for elderly users of interactive television. In *Proceedings of the 11th European Conference on Interactive TV and Video*, EuroITV '13, pages 55–58, New York, NY, USA, 2013. ACM.

[36] Philip J Hayes and Jaime Guillermo Carbonell. A tutorial on techniques and applications for natural language processing. 1983.

[37] Lode Hoste and Beat Signer. Speeg2: a speech-and gesture-based interface for efficient controller-free text input. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 213–220. ACM, 2013.

[38] Aseel Ibrahim and Pontus Johansson. Multimodal dialogue systems for interactive tvapplications. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, page 117. IEEE Computer Society, 2002.

[39] Takeo Igarashi and John F Hughes. Voice as sound: using non-verbal voice input for interactive control. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 155–156. ACM, 2001.

[40] Jeff Sauro. Measuring usability with the system usability scale (sus). `http://www.measuringu.com/sus.php`, 2011.

[41] Jeff Sauro. 10 things to know about the single ease question (seq). `http://www.measuringu.com/blog/seq10.php`, 2012.

[42] Jiepu Jiang, Wei Jeng, and Daqing He. How do users respond to voice input errors?: lexical and phonetic query reformulation in voice search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 143–152. ACM, 2013.

[43] Percy Liang. Talking to computers in natural language. *XRDS: Crossroads, The ACM Magazine for Students*, 21(1):18–21, 2014.

[44] Clifford Ivar Nass and Scott Brave. *Wired for speech: How voice activates and advances the human-computer relationship.* MIT press Cambridge, 2005.

[45] Marianna Obrist, Pablo Cesar, David Geerts, Tom Bartindale, and Elizabeth F. Churchill. Online video and interactive tv experiences. *interactions*, 22(5):32–37, August 2015.

[46] Gang Ren and Eamonn O'Neill. Freehand gestural text entry for interactive tv. In *Proceedings of the 11th European Conference on Interactive TV and Video*, EuroITV '13, pages 121–130, New York, NY, USA, 2013. ACM.

[47] R.B. Rumbolt and W.R. McIntyre. Universal remote control unit, September 27 1988. US Patent 4,774,511.

[48] Jeff Sauro and James R Lewis. Correlations among prototypical usability metrics: evidence for the construct of usability. In *Proceedings*

of the SIGCHI Conference on Human Factors in Computing Systems, pages 1609–1618. ACM, 2009.

[49] Nikhil Sharma. siri thinks i have two wives & other embarrassing voice interactions. *CHI 2015 Workshop: Embarrasing Interactions*, 1(1), 2015.

[50] Tony Sheeder and Jennifer Balogh. Say it like you mean it: Priming for structure in caller responses to a spoken dialog system. *International Journal of Speech Technology*, 6(2):103–111, 2003.

[51] Lisa Stifelman, Adam Elman, and Anne Sullivan. Designing natural speech interactions for the living room. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 1215–1220, New York, NY, USA, 2013. ACM.

[52] N. Tesla. Method of and apparatus for controlling mechanism of moving vessels or vehicles, November 8 1898. US Patent 613,809.

[53] Anton Treskunov, Mike Darnell, and Rongrong Wang. Active haptic feedback for touch enabled tv remote. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, pages 319–322, New York, NY, USA, 2015. ACM.

[54] Radu-Daniel Vatavu. User-defined gestures for free-hand tv control. In *Proceedings of the 10th European conference on Interactive tv and video*, pages 45–48. ACM, 2012.

[55] Radu-Daniel Vatavu and Ionut-Alexandru Zaiti. Leap gestures for tv: insights from an elicitation study. In *Proceedings of the 2014 ACM international conference on Interactive experiences for TV and online video*, pages 131–138. ACM, 2014.

[56] Andy Weir. Google announces android tv. `http://www.neowin.net/news/google-announces-android-tv`, 2014.

[57] Kent Wittenburg, Tom Lanning, Derek Schwenke, Hal Shubin, and Anthony Vetro. The prospects for unrestricted speech input for tv content search. In *Proceedings of the working conference on Advanced visual interfaces*, pages 352–359. ACM, 2006.