Applications of Semantic Technologies

Table of contents

Introduction

Interactions between humans and computers have become common place. Although for robots to live alongside humans they need to possess some form of emotional intelligence. In this project we look at how we can utilize the information harvested from natural language processing (NLP), to determine someone's sentiment. We especially look at how emotionally intelligent a machine can become. The goal of this project is to maximize the emotional quotient of a robot. Similar to the IQ the emotional quotient is a scale of how emotionally intelligent someone is. There first step would be a knowledge base of emotions from which we can infer the emotion from a given sentiment.

The next step was to have scenarios created using a modeling procedure. These scenarios are depicted by a model. The user can create a scenario, which adjusts his environment to be in a specific state, depending on the user's mood. To be able to define scenarios it was necessary to create a metamodel of the robots possible actions. The actions are then mapped to emotional values/moods from a knowledge base such as an ontology. Only a linear scale of emotions need to be characterized and the rest is inferred. Additionally, fuzzy sets can also be considered as an alternative and/or an addition to an ontology. One can even consider to integrate fuzzy sets with ontologies.

To validate the scenario, a NAO robot is used. The microphone and the

Software Development Kit (SDK) of the NAO is used to capture audio clips and to communicate these clips to the Google servers via a REST API. The REST response from the API (speech to text) would be used to send another request. The second request would return the overall sentiment of the previously sent text. The sentiment is then used by the NAO to adjust the current scenario (a scenario is the current state of all the IOT devices in the local environment).

Having a robot as an intelligent assistant e.g. a NAO at home will be like having a butler. He can run to different rooms, ensure everyone is safe and report anything suspicious. But, not only is the NAO interacting with the different devices inside of the house, he is interacting with the people that live there. For that, NAO needs emotional intelligence. For us to be able to make the NAO emotionally intelligent, we need a way for it to analyze and acts on signals e.g. the first step is converting sound waves into text. This is already one of the most difficult tasks for AI and falls into the AI-complete category. Not only is it necessary to identify which language, it is also necessary to identify the information correctly. The correct interpretation can be hard to find. Sarcasm among other language techniques is nearly impossible to detect. Each word in a sentence and its lexical meaning needs to be identified. After that, the semantics context needs to be determined. The NAO will not always interpret everything correctly. Giving the NAO fault tolerance and corrective mechanisms is essential. Of course, it is not necessary to have the lexical meaning of every word for the semantic analysis. It is rather the sentiment of each word that is needed. But since most systems run on commands, these commands need the lexical meaning. Having to solve the context and the sentiment increases processing. This has lead to centralized cloud solutions with light weight robot clients.

Conceptualization

We will dive deeper into more specific scenarios in which a robot uses various classification algorithms to determine the mood of the user. Not only will the machine classify the mood but it will also act upon the results. Imagine being able to have a personal assistant that, depending on how you feel, always knows what to do. In this project, scenarios are modeled by the user. The user will be able to model the scenarios from a metamodel. The robot can then execute a scenario after classifying the user's emotions. An example of a scenario would be modeling the lamps of an apartment and assigning different colors to those lamps depending on the mood. It would also be possible to connect individual colors to a mood. This brings us the specific approach.

Below we will describe six scenarios. These scenarios will clearly show what we wish to implement in our project. Each scenario has a use case description detailing what exactly it tries to achieve. The second part of each scenario describes the deliverables. These are the technologies/concepts that will be used/produced. The last part of each scenario are the key performance indicators. These are the values/variables that will be looked at to measure how successful our implementation is in fulfilling the described use case.

For our use cases we will follow the vertical transformation of data to knowledge. Our first use case will be concentrated on the mining of data and bringing this data the long way from having weak semantics up to our final use case, where it will have strong semantics. At the risk of being too vague, our first use case starts off simple, so that the other two use cases can build upon it.

Scenario 1: Voice to Text

In this first scenario we want to make it possible for the NAO to start and stop recording audio. The NAO needs to be aware when someone is talking to it in order to start recording. It also has to recognize when someone is done talking to stop the recording. The NAO has multiple built in microphones that can be used to record audio.

1.1 Use Case Description

This is a rather low level use case. It looks at transforming analogue data into digital data. Further, it looks at detecting (sound) start and stop events. The two actors in this use case are the NAO robot and the human who is interacting with the NAO. The purpose of this use case is to gather the necessary raw data for use cases 2 and 3 so that they can provide semantically meaningful information, knowledge and actions. Even though this is a low level use case its accomplishment is no small feat. Recording at the right time and reducing noise will be challenging.

The success scenario for this use case would be for someone to have a short conversation with the NAO. An example of this conversation:

1. The user initiates a conversation.

User: “Hello NAO, i’m home”

2. The NAO detects the word NAO using *ALSSpeechRecognition*

3. NAO starts recording audio using *ALAudioRecorder*
4. NAO uses *ALSTextToSpeech* to answer.

NAO: “Hello, how was your day”

User: “It was great!”

5. NAO detects the user is talking through *ALSSoundDetection*
6. As soon as the sound level reaches its signal mean again NAO stops recording
7. NAO responds with a predefined phrase and stops the session.

NAO: “I’m glad to hear that”

8. The raw data is saved and can be used for further processing depending on the users model defined in the other use cases.

Exceptions can occur when the NAO starts recording at the wrong time. The stop recording event needs to be triggered at some point. A timeout should be set to ensure that faulty recordings do not become to large.

1.2 Deliverables

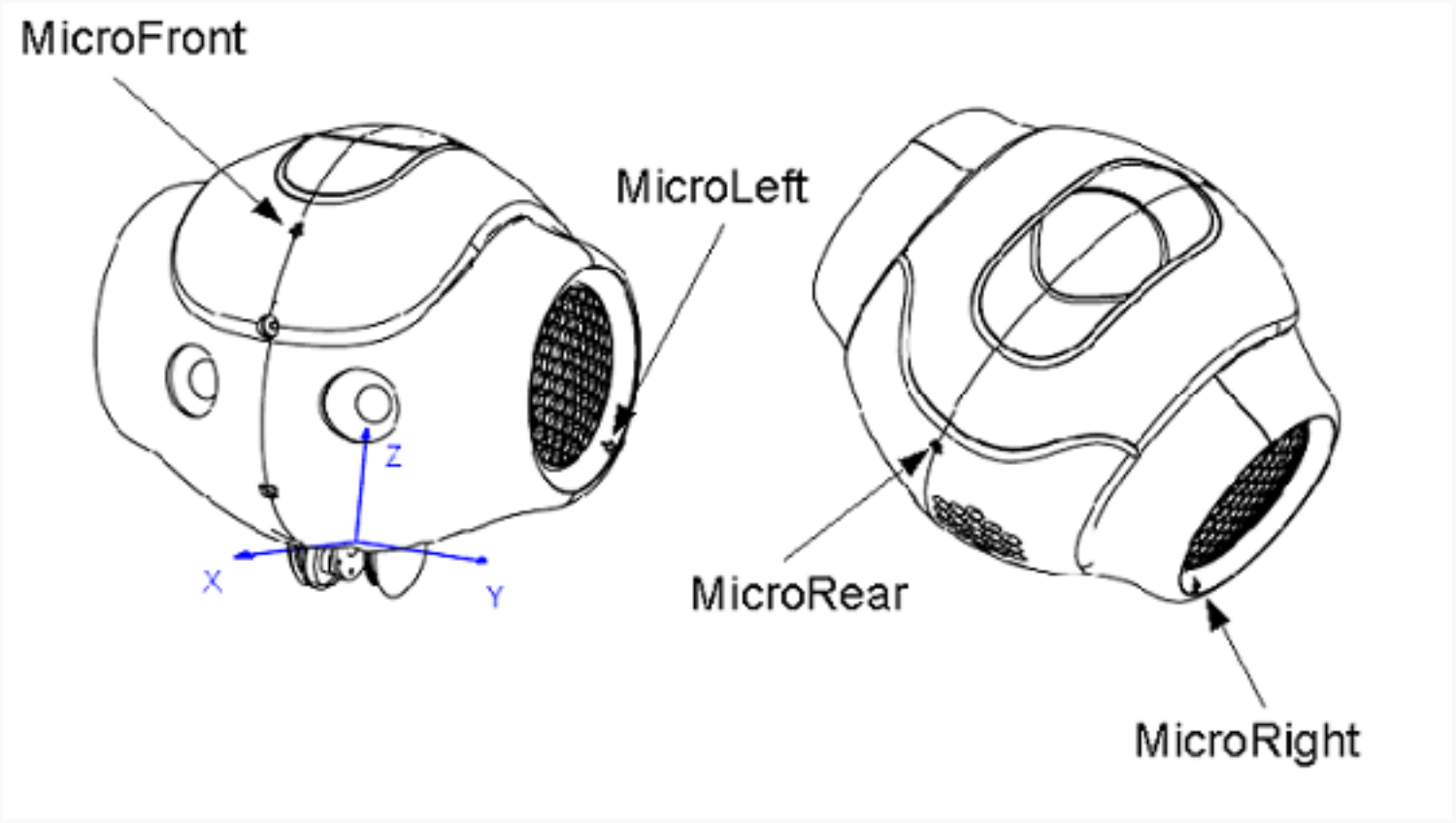
The deliverable we wish to present is a runnable program that will execute on our end device. This end device, the NAO, will be essential in gathering raw data. Without raw data, information and knowledge cannot be derived. The NAO robot comes with an SDK that we intend to use.

It offers a sound management library. This library has the functions necessary to record sound and detect sound events.

The most important parts are listed below and can be found on the NAO SDK site:

Name	Description
ALSSpeechRecognition	Detects predefined words in multiple languages.

ALSSoundDetection	Detects sound that differs greatly from the signal mean - peak based detection.
ALTextToSpeech	Lets the robot speak such that he can respond.
ALAudioRecorder	Provides audio recordings in WAV and OGG at 48000 Hz. Relies on Linux libraries SNDFile and ALSA.



1.3 Performance Indicators

Compare the recorded audio file to a script that has been written beforehand and spoken to the NAO bot.

Criteria	Description
Clarity	How well the person can be understood from the recording on a scale from 1 to 10.
Completeness	Check wether the speaker was cut of at the beginning or the end of the recording.
Length	Compare the length of the audio file to the time that

	was actually spent talking.
Signal to Noise	Check how strong the noise level is.
False Alarm	Count the times that the NAO starts recording without there being any intent to talk to it.

Scenario 2: Sentiment Analysis

This scenario looks deeply into a subject field of semantic technology, namely Natural Language Processing (NLP). This field has become very popular in the past years. It is being used in all sorts of new devices on the market. Devices such as the Alexa, Google home and Siri. These devices try to understand us and to talk to us. They may still have some difficulty in keeping up with the context of a conversation. Although soon Alexa could be your best friend from whom you would hear the newest gossip and get the latest fashion recommendations.

2.1 Use Case Description

This is a higher level use case. It builds upon the raw data gathered in the first use case. The challenges of this use case can be split into two major parts.

The first part is converting the audio file into text. The sentiment analysis can better be performed on text than it can on sound. In order for us to convert the audio file we need a machine learning algorithm that has been trained.

The second part would be using the text to figure out the sentiment of it. The baseline algorithm consists of:

- Tokenization
  - Deal with markup (xml, html)
  - Capitalization
  - Phone numbers, dates
- Feature Extraction
  - What words to exclude/include
  - Include all words or just the adjectives?
- Classification
  - Naive Bayes

    - MaxEnt
    - SVM

The above list also displays some difficulties with creating a functioning model. It is especially difficult to find a suitable dataset to train your model. Fortunately, Google provides an API for both speech to text as well as for sentiment analysis. The success scenario would consists of multiple requests that looks something like this:

1. It is possible to send a POST to https://speech.googleapis.com/v1/speech:recognize. This post includes metadata (config) about the audio file as well as a base64 encoding of the content.

```
{
"config": {
    "encoding": "WAV",
    "sampleRateHertz": 44100,
    "languageCode": "AT"
  },
  "audio": {
    "content":"U29tZSBieXRlcyBhcyBhIHN0cmluZyBiYXNlIDY0IGVuY29kZWQ="
  },
}
```

2. Google translates the audio content into the desired language specified. The server then responds with a result array that includes a transcript, the confidence and the words of the audio file.

```
{
  "results": [
    {
      "transcript": "Enjoy your vacation!",
      "confidence": 0.9,
      "words": [
        {
            "word": "Enjoy",
        },
        {
            "word": "your",
        },
        {
            "word": "vacation",
        }
      ],
    }
  ],
}
```

3. Using the result from step 2 we can send another request to google. This requests includes metadata about the encoding as well as the text to be analyzed.

```
{
  "encodingType": "UTF8",
  "document": {
    "type": "PLAIN_TEXT",
    "content": "Enjoy your vacation!"
  }
}
```

4. The servers respond with the overall document sentiment as well as the sentiment of individual sentences of the content. The magnitude is a sum of how positive and negative the document sentiment is. The score specifies how positive or negative the overall document sentiment is. A document sentiment with a neutral score can still have a high magnitude as positive and negative sentiments cancel each other out in the score.

```
{
  "documentSentiment": {
    "magnitude": 0.8,
    "score": 0.8
  },
  "language": "en",
  "sentences": [
    {
      "text": {
        "content": "Enjoy your vacation!",
        "beginOffset": 0
      },
      "sentiment": {
        "magnitude": 0.8,
        "score": 0.8
      }
    }
  ]
}
```

2.2 Deliverables

- NLP processing
  - Tokenization - Bag of words
  - Sentiment Analysis
- Knowledge base of emotions.

2.3 Performance Indicators

- As Bo Pang and Lillian Lee stated in their paper "If you are reading this because it is your darling fragrance, please wear it at home exclusively" (review by Luca Turin and tania Sanchez of the Givenchy perfume Amarige, in Perfumes: The Guide, Viking 2008). This sentence does not have any obvious negative words but its overall sentiment is negative. It would be possible to compose a list of these seemingly natural sentences and see how well the system performs. These sentences can be composed increasingly more difficult. Of course the expected result should be known beforehand, thus we can compare the expected with the observed.
- Compare speech from the audio file to the converted text.
- Emotional Quotient

Scenario 3: Smart Home controller

In the third step we want to bring all concepts together and create a working prototype that can change the room atmosphere depending on things that are said in the room. This means that we will implement a service, which can record human speech and transform the audio into a text file (scenario 1). This text will then be semantically analyzed through tokenization and sentiment analysis to extract moods and emotional intensity (scenario 2). Finally, in scenario 3, we will use this input and change the environment according to the atmosphere in the room (e.g. change lights and colors).

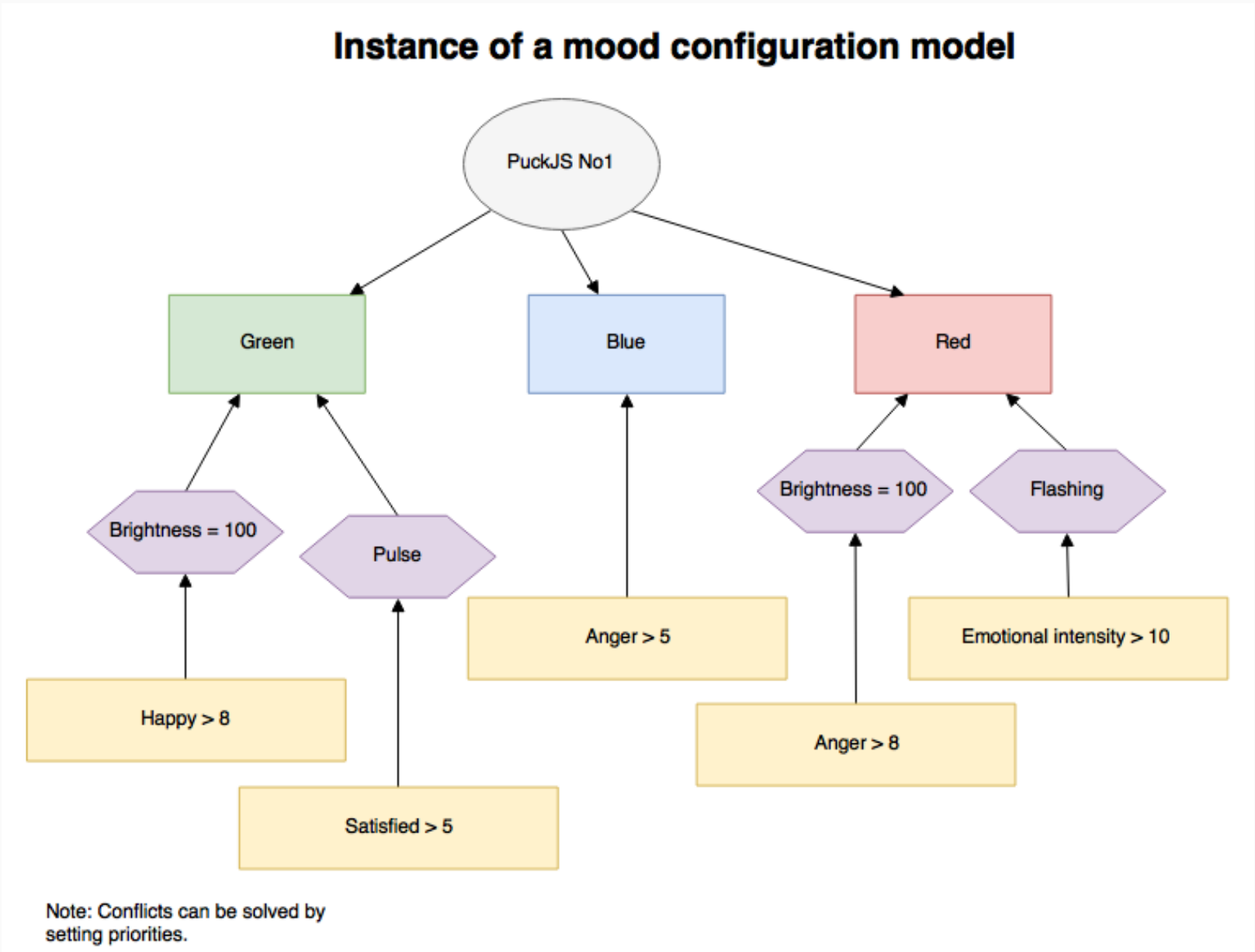
3.1 Use Case Description

To make it possible for someone to control their own home, a modeling language needs to be created. Through the modeling language the user will be able to control their smart home. We are talking about a smart home in which the lights can be adjusted depending on a certain criteria/emotion that the user can predetermine. This scenario definition needs to be recorded in some way, such as in a model. Using a model to describe a smart home will enable a lot of automation and will reduce setup time. Integrating different modeling procedures will let the user start up quickly and control their smart home however they desire.

The users need a language i.e. a metamodel through which they can describe the relationships and entities that are in his smart home. In other words, the user needs a metamodel that lets the user create a model, which can control not only his smart home but also lets the user describe emotions inside of the model. The metamodel should define an abstract conceptualization of the

objects in a smart home and of human emotions. By giving the user the possibility of creating models that can describe facts as well as emotions we are able to give a smart home emotional intelligence. This is limited to the extent that the user can only predefine specific scenarios and a model instance will not learn (from) the behavior of the user.

This system should be flexible to some degree there must be a way to define certain mood configurations (e.g. when the emotional intensity is above a certain threshold the lights turn red). These mood configurations should not be hard-coded, but be standardized, human readable and ideally visualizable. Therefore modelling techniques are suited to realize this requirement. A metamodel should provide a basic outline on how a mood configuration must look like. On the basis of that metamodel, any user can create a model that represents how they want their moods and emotional profiles to be handled. Once a user has created a model, that model can be exported and be used by our service.



Since this service will essentially be a home automation software it should be highly compatible, extensible and also have remote access. This is why a web service is suited for this task. The web service will control the text-to-speech

module and the sentiment analysis from scenario 1 and 2. The web service must also import the mood configuration model (created by a user).

Now, based on the sentiment analysis and the mood configuration model, the system needs to change the environment (e.g. color of the lights). To trigger this, a rule engine will be used. The rule engine gets the interpreted sentiment analysis as an input as well as the exported mood configuration model. The rule engine has a rule for each possible element of the model. If there is a match (e.g. emotional level is above a certain threshold) the rule fires.

When a rule from the rule engine is executed it needs to trigger a change in color of the lights. To do this the rule engine can call a script which then sends out a wireless signal. A lamp will listen for signals and will change its color accordingly.

Proposed software stack:

- **Operating system:** Any common Linux distribution will do. Linux is open-source, secure, free and can easily be tested locally. If possible the software can the be ported onto the NAO, since the NAO also runs on Linux
- **Models:** ADOxx can be used to create a metamodel as well as the actual models. It also supports XML export.
- **Engine/Web service:** NodeJS can be used. NodeJS is a JavaScript Runtime which is based on Chrome's V8 JS engine. It is event-based, non-blocking I/O, lightweight, easily extensible and supports many libraries.
- **XML-Import:** The web service / rule engine needs to analyze the mood configuration models and therefor they need to be imported. The already in NodeJS included library fs (FileSystem) can be used here. If needed an XML parser can additionally be used to read and interpret the model.
- **REST-API:** To make our program easily remotely accessible a REST-API will be implemented. Express is a very common way to realize REST-Services with NodeJS. It is an easy-to-use, very established library which is also quite powerful.
- **Rule engine:** As a rule engine node-rules can be used. Node-rules defines facts (JSON objects) and then executes rules based on these facts. Rules always consist of a condition and a consequence.
- **Lights:** To validate this prototype at least one actual lamp is required which can glow in different colors. Therefor puck.js can be used. It is an entirely autonomous device which has Bluetooth-support, a button and also an LED which can glow in three different colors. It can also be programmed in JavaScript which will synergise well with our JavaScript-

based main program.
- **Bluetooth:** Finally, to communicate wirelessly between the puck.js and the web service Bluetooth will be used as communication protocol. The puck.js supports Bluetooth natively. The system where the Linux distribution is running on need to have a hardware Bluetooth module, and the NodeJS application can use the library node-bluetooth to control the LED on the puck.js.

3.2 Deliverables

- An ADOxx metamodel which allows the creation and visualization of mood configurations
- At least one ADOxx model derived from the metamodel which will be used to test the service (XML export of the model)
- A working web service that includes
  - a REST API to control the service (text2speech and sentiment analysis) and imports the ADOxx model
  - a rule engine that gets the sentiment analysis as input and fires rules according to the ADOxx mood configuration model
  - a controller that can change the color of the lights when a rule fires.

- Modeling Procedures
- Graphrep

3.3 Performance Indicators

A fully working system: Human speech is automatically detected and recorded. According to the mood and emotional intensity the lights in the room will change colors. In order to test the success and performance of the prototype, test subjects can use the prototype and evaluate the following criteria:

Criteria	Description
Usability/Performance	The test subjects can rate the usability/performance of the prototype. Do the test subjects feel like the lights changed according to the mood of their conversation?
Speed	How long does it take for input to be evaluated and a color change to be triggered?

Overall usefulness	Do the test subjects feel like the prototype provides value? Do they like the functionality?

Development

Deployment

Conclusion

We have created a validation environment that checks whether or not The NAO robot is emotionally intelligent in his actions. If the validation environment runs multiple test cases and each of them succeeds then the validation method would be a success. The test cases should not require humans to monitor the system but a predefined test set with an estimated sentiment polarity should be used to see if the system is operational. Further, the system should validate the resulting state of the IOT devices with the representation in the Model.

We are not too far away from having a fully functional robot that can interpret our emotions. The difficulty comes in having to ensure that the interpretation is correct. In the future, it would be interesting to see what other devices in our environment could be controlled by our emotions. IOT in the future allows for everything to change without the user directly interacting with the system. The NAO bot could detect if the person was cold and adjust the heating.