ComfiCube: Crafting a Healthy Indoor Environment

Manan Rai
Department of Computer Science
Stanford University
mananrai@stanford.edu

Abstract

Millions of people are affected by workplace stress. What adds to the psychological pressure of performing at your best is the physical strain on the body, caused by lack of proper posture, not taking adequate breaks, or perhaps not sitting often enough. The psychological part of this well-being, moreover, can often be countered by increased self-awareness. ComfiCube is a chair that could help users achieve a healthier indoor environment. ComfiCube is a cube-shaped chair that uses a single camera to identify users and navigate to them, actuated by a mechanism that determines when it is best for users to sit. ComfiCube helps users introspect by mirroring their emotions. This paper details systems that identify the location of the user in the map of an indoor environment, as well as an Emojify Network that listens to conversations and predicts an emoji mirroring the sentiment in the conversation.

1 Introduction

The aim of the project is to build furniture that automatically intervene and help with humans' comfort. Here I propose the design for **robotic chairs** that aid in achieving this objective by **being able to navigate to users**, as well as **reflecting the current mood evidents from users' conversations**.

2 Design

Consider a cubical chair with LED displays on all 4 lateral sides. Similar designs include cube chairs (that are not robotic). The displays are used to impart a sort of personality to the chair, and they show an emoji-style face, reflecting the mood inherent in users' communication. Moreover, these chairs can navigate to the user whenever deemed fit from sensory considerations. Note that these suggestions are discussed considering a model restricted to a single room.

Figure 1: Cube-shaped Chairs

Manan Rai, 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Related Work

3.1 Sentiment Classification

Sentiment Classification is an important Machine Learning problem that is related to the Emojifying task we have at hand. There is not a lot of labeled sentiment-classification data that represents conversations, and most models learn by analyzing tweets. Note that emoji's can be more expressive and subtle than discrete sentiment classes, which makes this model better suited to our application. Moreover, our objective is to be able to display emoji's on the LED screens, and although a sentiment classifier with a sentiment-to-emoji layer on top could achieve similar results, the additional layer may offer an unnecessary computational overhead that can add up very quickly as the product is used.

4 Datasets

4.1 Predicting Emoji from Text

The model is trained on two different datasets with significantly different distributions of data. One has relatively simple, short sentences; the other has long, complex sentence structures and questions generated by communicating with the **PopBots**. Because of the disparate nature of the data, the same model does not generalize well to both forms of data. Trained separately, however, the same model achieves better results. I initially started with a third dataset as well, with 400,000 tweets.

The dataset with short sentences has 188 hand-labeled sentences with a 150/38 train/dev split. The PopBots dataset has 104 hand-labeled conversations with a 78/26 train/dev split.

4.2 Identifying Objects and Persons in Images

We use the 2017 MS COCO dataset with 118,000 training images, and use the 5,000 dev images for validation and testing.

5 Methods

5.1 Parsing Communications and Emojifying them

In order to achieve the goal of having the LED displays on the chair mimic the mood, we need to understand users' conversations. We can handle this task using the following process - we listen to users' communication and periodically run an emojifying neural network on the conversation. To make it easier for the network to understand features, we have an initial layer that converts speech into text using the `SpeechRecognition` python module.

For the emojifying network, we use an LSTM model that takes as input word sequences and outputs an appropriate emoji. The LSTM has the following architecture:

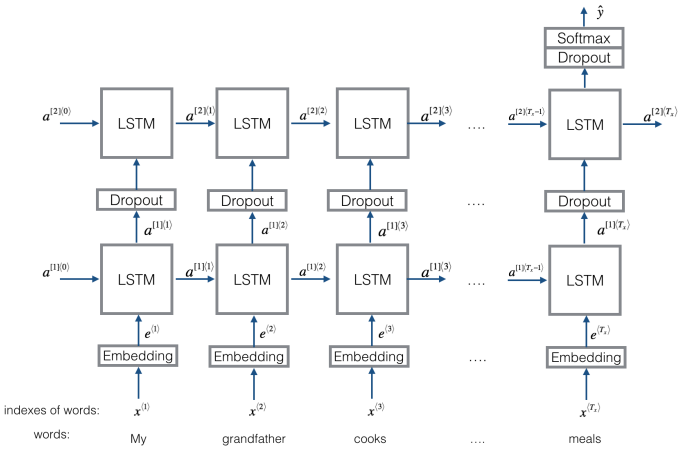

Figure 2: LSTM Architecture

and the Embedding Layers are:

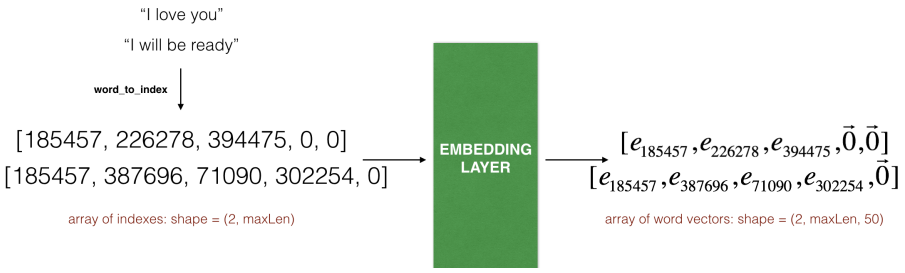

Figure 3: Embedding Layer

We use GloVe vectors to encode words.

5.2 Navigating to Users

We break down the task of navigating to users into two parts - identifying the location of the user, the chair, as well as obstacles around the room; and the actual navigation. This detection task is conducted using a camera installed at a corner of the room.

In order to identify the location of persons and objects, we conduct semantic segmentation of images using a Mask R-CNN trained on the MS COCO dataset. The Mask R-CNN is known for effective semantic segmentation, so this gives us a mask and bounding box for every person an object in a provided image.

The architecture of the model is shown below:

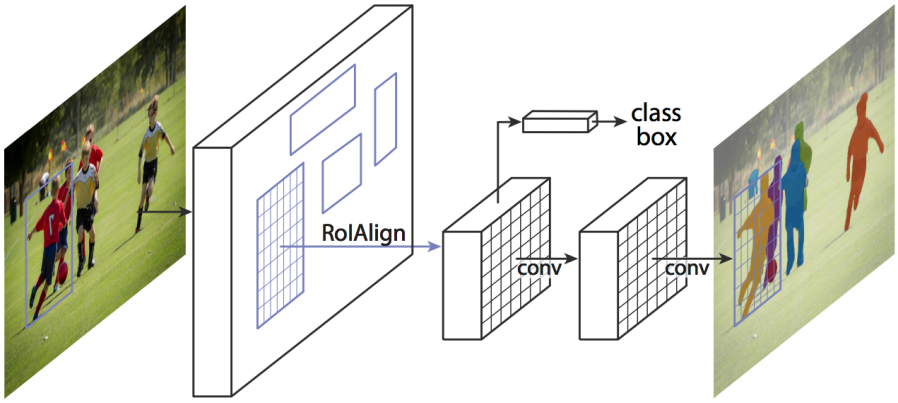

Figure 4: Mask R-CNN Architecture

Note that after we get the bounding box for every person in the picture, we use certain properties of the camera and its configuration to extract information regarding the users' position on a hypothetical 2D-floormap of the room. The information we need about the setup includes the height (h_o) of the object (for instance, approx. 6 feet for a person), the height (h_c) of the camera, the focal length (f) of the camera, the angle of depression (θ) of the camera, the DPI (dots per inch) in the image, and the height (H) of the image itself. Now we use the bounding box to find the height (h_i) and width (w_i) of the object in the image. Then the horizontal distance of the object from the camera in the real world is given by

$$d_{pixels} = \frac{f \times h_o \times H}{h_c \times h_i} \times \frac{1}{\cos(\theta)}$$

and

$$d_{feet} = \frac{d_{pixels}}{\text{DPI} \times 12}.$$

Now, let the position of the current object of interest be (x, y). Then

$$x_{pixels} = \text{(x-coordinate of mid-point of object)} - \text{(x-coordinate of mid-point of image)}$$

and again,

$$x = x_{feet} = \frac{x_{pixels}}{\text{DPI} \times 12}.$$

Consequently, the y-coordinate is given by

$$y = \sqrt{d_{feet}^2 - x_{feet}^2}$$

since d_{feet} is the distance between the camera and the mid-point of the object, and x_{feet} is the x-offset between the same two points, and therefore, x_{feet}, y_{feet}, and d_{feet} for a right-triangle with hypotenuse d_{feet}. Repeating this process of finding (x, y) for every object, we can build a 2D-map of the floor of the room, mapping all objects, with the point on the floor corresponding to the camera at the origin.

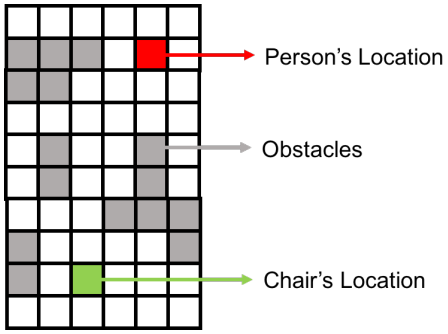

Figure 5: A Sample 2D Floor Map

Now we have a map with white grid locations representing empty, navigable spots on the floor, and grey cells marking obstacles. We can use a path-finding algorithm such as A* (or a variation such as the bidirectional A*) in order to calculate a path from the chair to the user. The actual navigation can then be carried out by the hardware of the chair. Note that this process of navigation is actuated by a separate mechanism that subliminally tracks the user's needs and identifies *when* the user should sit down.

6 Results

6.1 Emojify Network

The speech-to-text pipeline has not been formally tested locally, however since it is a production-ready module that is used as an industry standard, we can rely on its accuracy.

In order to quantify results, we use a continuous grading scheme. For every test example, we define a set of correct emoji's, each of which is assigned a different multiple of 0.25 as its score. For instance, we can use the following scale: 😀 gets 1 point, 🙂 gets 0.75 points, 😊 gets 0.5 points, and 😌 gets 0.25 points.

The results achieved by the LSTM are summarized below:

Dataset	Accuracy
Short Sentences	74.34%
PopBots Data	80.77%

Some sample results of the emoji prediction algorithm are shown:

```
prediction: not feeling happy😞
prediction: I love you so much❤️
prediction: let us play ball⚽
```

Figure 6: Predictions for sample data from the 'short sentences' dataset

```
prediction: Did that help you to find something good (or at least funny) about the situation😉
prediction: Perhaps😏
prediction: I'm glad Would you consider trying this strategy such as finding a joke in the future👍
prediction: Maybe😏
```

Figure 7: Predictions for sample data from the 'PopBots' dataset

6.2 Person Tracking

Tested on select images from the 2017 MS COCO Val set, the Mask R-CNN achieved the following results for the bounding box calculations:

Average Precision (IoU > 0.5)	0.683
Average Recall (IoU > 0.5)	0.487

Sample output of the Mask R-CNN is shown below:

Figure 8: Predictions for a sample image from the MS COCO 2017 dataset

7 Discussion

7.1 Different Datasets

The Emojify Network has two sets of weights better suited to different data distributions. It is wise to have a program that distinguishes input data and uses the weights that are better suited. This distinguishing could be done on the basis of:

- length
- number of sentence fragments

If either of these is high, the input is better suited to the PopBots weights. If both are fairly low, then the first weights may predict better.

7.2 Theoretical versus Practical Accuracy

It is important to note that written conversation has typographical errors because of which input words cannot be found in the `word_to_index` vector, and this causes errors. These errors would not happen when we use an initial speech-to-text layer since the `SpeechRecognition` module already applies auto-correction to words.

7.3 Relevance to the PopBots Project

Since the actual use case involves understanding conversations happening inside a room, the PopBots dataset is really close to the target data distribution. Careful observation of the conversations led me to come up with certain use cases where the Emojify Network may be useful for the PopBots project.

- The PopBots currently apply an algorithmic approach to communication, and emojis/sentiment analysis can be used as a feedback mechanism. Consider, for instance, Sir Laughs a Bot. When he asks the user for a joke, the user may respond, instead of making a joke, by saying that their situation is way to serious for them to be able to make a joke. The bot can use emoji prediction to identify if the response seems to be a joke or a serious comment, which can then inform the route of the subsequent conversation.

- Prediction of emoji's can also be used to identify the trend of the user's responses. If the responses go from negative to increasingly positive ones, then it suggests that the bots are doing a good job.

- The feedback can also be with respect to the bot's choice of words and phrases. This can be incorporated into the base system by adding to the cost function an error for the emoji/sentiment predicted. A Reinforcement Learning framework that seeks to make the emoji/sentiment predicted more positive can be used to polish the bot's side of conversations.

8 Next Steps

Further parts of the project include creating a system for the chair to navigate to the user. We also need to build on the initial sensory infrastructure that asks chairs to navigate to a user. We need to obtain a compatible camera and experiment with the Mask R-CNN running on video data in real time.

9 Future Work/Possible Extensions

An extension that could be implemented is implementing trigger word detection (such as "Alexa" for Amazon Echo) using which the user can establish communication with the chair, or even call the chair towards themselves.

Another extension may be looking into the U-Net architecture, which is also known for semantic segmentation, and comparing its performance with that of the Mask R-CNN. Note that the U-Net only carries out segmentation, and unlike the Mask R-CNN, does not predict a bounding box. So this will involve some extra calculations to predict the user's position in a 2D map.

10 Code

`https://github.com/mananrai/ComfiCube`

References

Klaus Greff, Rupesh K. Srivastava, Jan Koutnık, Bas R. Steunebrink, Jurgen Schmidhuber. LSTM: A Search Space Odyssey. arXiv:1503.04069v2.

Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick. Mask R-CNN. arXiv:1703.06870.

Mask R-CNN Implementation retrieved from `https://github.com/matterport/Mask_RCNN/tree/master/mrcnn`.

Emojify app inspired by CS 230, Spring 2018,(Deep Learning: Sequential Models, on Coursera) and `https://github.com/cryer/Emojify/`.