TUM Data Innovation Lab
Department of Mathematics
Technical University of Munich

Emotional states and personality profiles in Conversational AI

C. Bauerhenne, A. Gammoudi, M. Moussaoui, J. Reichelt, J. Wang

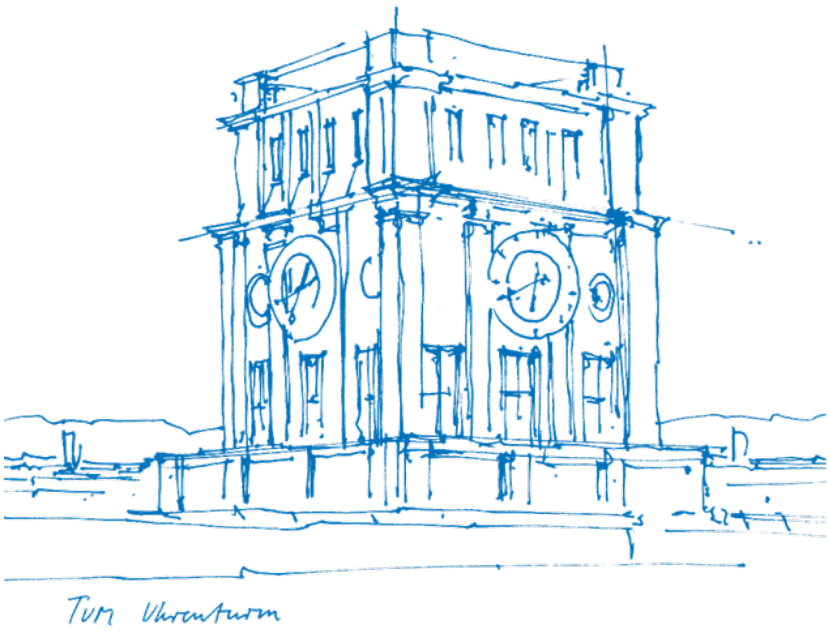
Sponsored by
Steering Lab by Horváth & Partners GmbH

Project Lead
Dr. Ricardo Acevedo Cabra

Scientific Lead
Olena Schüssler, Dr. Inna Vasylchuk

Co-Mentor
Michael Rauchensteiner

Submitted
22.07.2020



Abstract

One of the most fundamental challenges in artificial intelligence is enabling the machine to converse with humans using natural language. Some years ago, intelligent personal assistants were developed. Today, a new conversational system has emerged: a social chatbot aims not primarily at solving all the questions a user might have, but rather at being able to establish an emotional connection with the user. This is achieved by satisfying the user's need for communication and affection. We present a chatbot that detects the Six Basic Emotions by Ekman using a hybrid approach including LSTM. It can also adapt any personality defined by the *OCEAN* model. Based on the user's emotion and its own personality, the chatbot reacts in an emotionally meaningful way.

Contents

1 Introduction

Motivation

Conversations between humans and machines are nowadays commonplace and chatbot interactions replace human services more and more. Humanizing machines is thus of great importance, as a first step to improve user engagement and in the future to eventually give intelligent machines a moral compass [1]. Especially in customer service it is easy to see how an affective assistant that can react to and learn from emotions enhances customer satisfaction compared to an emotionless machine.

One of the most fundamental challenges in artificial intelligence is enabling the machine to converse with humans using natural language [2]. Starting with key-word based conversational systems such as Eliza and Parry in the 1960s, moving to task-completion systems in the 2000s, intelligent personal assistants such as Siri, Cortana, and Alexa were developed in the 2010s. Nowadays, conversations between humans and machines are common and social chatbots like XiaoIce have emerged as new conversational systems. Google recently presented the seemingly human-like chatbot Meena [3] And while text-based sentiment analysis, which classifies emotions as positive, negative or neutral, is a well established field in natural language processing [4, 5], emotion detection is far from similar success and ubiquity [6]. The surveys [6, 7, 8] provide an overview over existing machine learning approaches for text-based emotion detection. In particular, hybrid models using deep learning, traditional machine learning and key-word based approaches achieve accuracies between 57% and 88%, the latter being comparable to human accuracy levels [9].

Project goal

Our goal was to develop an emotionally aware chatbot that can detect accurately, in real-time, emotions in short text messages. Moreover, to be categorized as an affective personal assistant, the ability to express sentiments is also required. Ultimately, we developed a bot that can differentiate between the six basic human emotions *joy, surprise, anger, sadness, disgust, fear*, and *neutral*. Moreover, it can adapt any personality defined by the personality model, the *OCEAN model*, and changes its mood according to its personality and the user's expressed emotions. The chatbot's emotion detection component, achieved a classification accuracy of 88% when using a Bert-large-uncased model for multi-labeled sentences on Semeval18's test set. Using RNN models with LSTM, BiGRU, and GRU, we achieved an accuracy of 86%, 82%, and 81% respectively on Unified Dataset.

Outline

In Chapter 2, we discuss different emotion models, define two chatbot personalities, and explain how they interact to determine the chatbot's reactions. In Chapter 3, we present suitable text-based datasets, our emotion detection approaches, and finally, our hybrid approach involving an emotion detection component using an LSTM model and a key-word based approach. In Chapter 4, we show how we built the chatbot in the open-source machine learning framework Rasa [10], which is a tool to implement voice or, in our case, text-based assistants. It provides a platform to easily combine language understanding, processing, and text generation modules as well as an interface for interactive learning and testing. However, most of our research can be used independently of the platform.

2 Psychological models

2.1 Modeling emotion

There are two fundamental approaches to model emotions in psychological and affective computing literature: categorical and dimensional models. Categorical emotion models assume that all humans have a given discrete set of basic emotions. Dimensional emotion models represent emotions in a continuous space which is usually described by the three dimensions *valence*, *arousal*, and *dominance*.

Research following the categorical approach is based on determining which emotions are basic. The most common categorical model, Ekman's *Six Basic Emotions*, distinguishes between *joy*, *anger*, *fear*, *disgust*, *sadness*, and *surprise* [11]. There are many extensions of this model, as well as other categorical models [12, 13, 14]. However, the categorical approach has two downsides: On the one hand, the idea itself to universally classify emotions is challenged, as there are for example intra- and intercultural differences in the way how emotions are valued and expressed [15, 16]. On the other hand, categorical models do not take into account the intensity of emotions.

The dimensional approach originates from factor analyses showing a relationship between emotional responses and the three underlying so-called *VAD* dimensions: *valence* (how positive or negative an experience feels), *arousal* (how energized or enervated the experience feels), and *dominance* (how controlling or submissive it feels) [17]. In [18], more than 150 discrete emotions have been embedded into the three-dimensional *VAD* space (Figure 4.5). Therefore, the three *VAD* values and discrete emotions were surveyed simultaneously. Then each categorical emotion is identified with the centroid of its corresponding categorical cluster.

Thus, not only can a wide range of categorical models be directly embedded into the *VAD* space but the dimensional approach also includes by definition more complex emotions as well as intensity. Therefore, the dimensional approach turns out to be more flexible compared to the categorical approach.

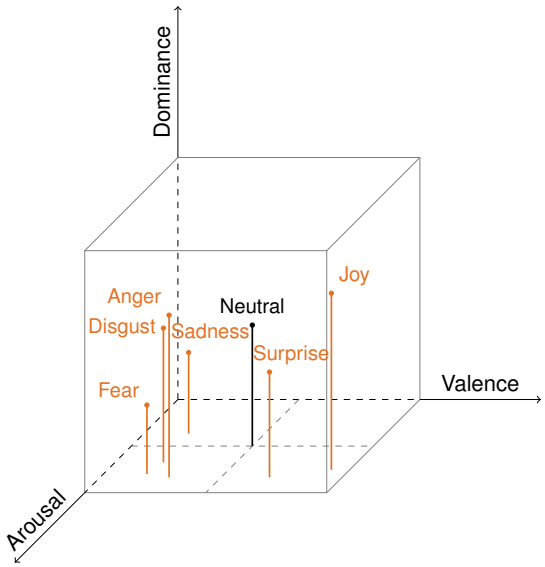


Figure 2.1 VAD embedding

Aiming at a chatbot that is easily adaptable to various domains with possibly distinct relevant discrete emotions, it seems promising to model emotions via the dimensional approach, i.e. to learn *VAD* values from text. However, on the one hand, section 3.1 will reveal that significantly more text data is available with labels based on the *Six Basic Emotions* model than on the *VAD* space and an approach to learn *VAD* values from data with categorical labels based on [19] was unsuccessful due to notation obscurities in the paper. On the other hand, a categorical emotion detection model does not severely restrict the chatbot's adaptability as long as its reactions are easily adaptable to a wide range of detected categorical emotions. Therefore, the emotion detection model in Section 3.2 is trained using data labeled with the *Six Basic Emotions* plus "neutral" and the chatbot's reaction is modelled based on the dimensional approach.

2.2 Personality profiles

In psychological literature, the most commonly used model to describe personality is, to the best of our knowledge, the *OCEAN* model: it represents personality using the the five dimensions *openness*, *consciousness*, *extroversion*, *agreeableness*, and *neuroticism*. The five dimensions were observed in many factor analyses and are found to be generally stable over time. We represent the values for each dimension on a scale from $[-1, 1]$ (see Figure 2.2).

	-1		1
Openness:	conventional, prefers routine	vs.	curious, independent
Consciousness:	impulsive, careless	vs.	dependable, organized
Extroversion:	quiet, reserved	vs.	outgoing, warm
Agreeableness:	critical, unccoperative	vs.	trusting, empathetic
Neuroticism:	calm, even-tempered	vs.	prone to negative emotions

Figure 2.2 OCEAN model

Based on the *OCEAN* model, our chatbot can adopt a wide range of personalities. For demonstration purposes we fix two different ones: the so-called “empathetic” personality with a high *agreeableness* value and the so-called “apathetic” personality that shall appear more neutral and business-solving. [20] presents a commonly used mapping from the *OCEAN* model space to the *VAD* space. Using this mapping, we assign the following *OCEAN* and *VAD* values to our two personalities:

	Openness	Consciousness	Extroversion	Agreeableness	Neuroticism	Valence	Arousal	Dominance
empathetic	1	0.5	0.5	1	0	0.695	0.45	0.315
apathetic	-0.5	1	0	0	-1	-0.19	0.495	0.045

Figure 2.3 Personality values

2.3 Mood-dependent responses

Our chatbot shall respond, dependent on its mood, in a friendly, neutral or mean manner. Based on [21], we model mood as a function of personality and detected user emotions: Our chatbot’s initial mood coincides with the *VAD* value of the bot’s personality. After each user message, either a basic emotion or "neutral" is detected. If a basic emotion is assigned to the latest user message, the mood is updated to a convex combination of the current mood and the detected emotion (Figure 4.5). The weight depends on the personality’s *neuroticism* value:

$$\text{mood}_{\text{new}} = \text{mood}_{\text{current}} + 0.90\,(\textit{neuroticism} + 1)\,\text{emotion}.$$

For example, the apathetic bot moves slower in direction of a detected basic emotion than the empathetic bot, as its *neuroticism* value characterizes it to be more even-tempered. If the latest user message is classified as "neutral", the bot’s mood moves back toward the *VAD* value of the personality using the same weights as above.

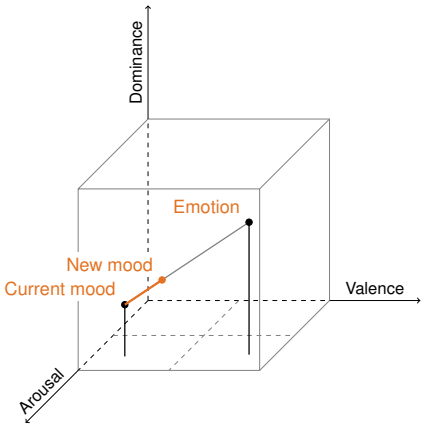


Figure 2.4 Mood embedding

Finally, in order to determine whether the bot responds in a friendly, neutral or mean manner, the *VAD* space is divided into corresponding three regions based on [22, 18]. The bot responds according to the area that its current mood is located at. For example, the empathetic personality is located in the friendly region and its mood moves through the *VAD* space rather quickly. The apathetic personality is located in the neutral part and its mood barely leaves the neutral area.

3 Natural Language Understanding

3.1 Datasets

Since datasets are indispensable, the selection of suitable datasets became the first step of our project. We have conducted in-depth research in datasets for emotion detection by searching through the data platform Kaggle, and reading papers about currently used ones, finally finding data from semantic evaluation competitions. After collecting a sufficient number of datasets, we compared their contents, and selected several candidates for the follow-up model training. In the training part, we analyze the behaviors of each candidate, which helps to make the final decision. In the following, we give a brief comparison among collected datasets, a detailed introduction on two representative ones in this field, and also a unified dataset which is the combination of several famous ones.

3.1.1 Dataset comparison

An overview of datasets for emotion detection tasks is provided in Table 3.1. These datasets are different in sizes, label numbers, topics, text types and annotation schemas. Therefore, it is necessary to identify the relevance of each one to our task and select a suitable one comprehensively.

Following the annotation schema, the list of datasets can be split into two types, categorical [23] and numerical. Categorical datasets use discrete categories as labels, which are determined by emotion models such as Ekman's basic emotions. The majority of categorical datasets contain the six basic emotions *joy*, *anger*, *sadness*, *surprise*, *fear* and *disgust*. Some categorical ones subdivide emotions further and add labels, such as *love*, *guilt* and *optimism*. These detailed emotions are more accurate, but also difficult for the model to detect. Besides, some emotions among them are very ambiguous for humans. Therefore, a dataset with a limited number of emotion labels is desired for training a chatbot. In contrast, numerical datasets [24] use continuous numbers to represent the emotion of each data point. For example, *VAD* values assign to each sentence a three-dimensional vector, which is described in 2. With the development of various kinds of emotion modeling approaches, some datasets adopt both categorical labels and numerical intensity values [25] for a better representation.

Granularity and topic are another two critical characteristics for the selection. The type of granularity controls the language style in the dataset. Social media posts, such as tweets [26] and Facebook posts [27], are shorter and colloquial. People tend to use fewer sentences and more hashtags to express their emotions. On the contrary, descriptions, news headlines [28] and whole sentences [29] are more formal. Instead of using words that can deliver emotions explicitly, they tend to implicitly include emotions in the context. Therefore, without the entire context and its language style, the emotion contained in the latter has to be learned from the context. Furthermore, conversations are another useful type for our task. They consist of dialogues between two people, and each sentence in the conversation is labeled with an emotion category or a *VAD* value. Since our goal is to converse with humans, dialogue data is most suitable for us.

The topic provides supplementary descriptions of the dataset. For example, some collect sentences in literature or fairytales [30], which possess fewer overlapping features with daily conversations, and thus result in poor performance.

To exploit deep learning models in emotion detection, the size of the dataset is also very important. Small datasets cannot thoroughly train the model [31]. However, some large ones also face the problem of imbalanced labels and insufficient data quality. According to our observation, the number of sentences with label *surprise*, *disgust* and *fear* are often relatively small compared with other labels in most datasets. This imbalance will lead to a poor prediction of sentences with these labels.

At last, we have to consider the number of labels for each sentence. Emotions in sentences may be interpreted differently by different people, or in a different context. As a result, some datasets assign multiple labels. Nevertheless, some labels in one sentence may be contradictory, and it is also difficult for models to decide which labels to take in the prediction period. Therefore, the tradeoff between training complexity and the number of labels should be considered.

In addition to training and validation datasets from the Internet, we also generate a small test set to help us choose a suitable model. This small one contains 60 sentences in total for 6 emotions, which are *joy*, anger, *disgust*, *fear*, *sadness* and *surprise*. Each emotion has 10 corresponding sentences written by ourselves. In order to test the performance of our model in real conversations, 60 sentences are imitated possible dialogue that might appear during chatting with a bartender.

Dataset	Granularity	Annotation	Size	Topic	Label	Source
Affective Text	Headline	E+V	1,250	news	multi	[28]
Blogs	Sentence	E+ne+me	5,025	blogs	single	[29]
CrowdFlower	Tweet	E+CF	40,000	general	single	[32]
DailyDialog	Dialogue	E	13,118	multiple	single	[33]
Electoral-Tweets	Tweet	P	4,058	elections	single	[26]
EmoBank	Sentence	V+A+D	10,548	multiple	single	[24]
EmoInt	Tweet	E-DS	7,097	general	single	[25]
Emotion-Stimulus	Sentence	E+shame	2,424	general	single	[23]
Gb-valence-arousal	Facebook	V+A	2,895	questionnaire	multi	[27]
Grounded-Emotions	Tweet	HS	2,585	weather	single	[31]
ISEAR	Description	E+SG	7,665	events	single	[34]
Tales	Sentence	E	15,302	fairytales	single	[35]
SSEC	Tweet	P	4,868	general	multi	[36]
TEC	Tweet	E$\pm$S	21,051	general	single	[37]
Empathetic Dialogue	Dialogue	P+SG+ED	24,850	general	single	[38]
EmoContext	Dialogue	EC	38,424	general	single	[39]
SemEval-2018 (task 1)	Tweet	P+M	7,902	general	multi	[40]

Table 3.1 Dataset Comparison. Ann. refers to the following annotation schemata: [E] Ekman: *anger*, *disgust*, *fear*, *joy*, *sadness*, *surprise*, [P] Plutchik: *anger*, *disgust*, *fear*, *joy*, *sadness*, *surprise*, trust, anticipation, [CF] enthusiasm, fun, hate, *neutral*, love, boredom, relief, empty, [DS] *disgust*, *surprise*, [JS] happy, sad, [V] valence, [A] arousal, [D] dominance, [SG] shame, guilt, [$\pm$S] positive *surprise*, negative *surprise*, [ne] no emotion [me] mixed emotion and Availability refers to the following, [M] love, optimism, pessimism, [EC] happy, sad, angry, others, [ED] excited, proud, annoyed, grateful, lonely, terrified, impressed, hopeful, confident, furious, anxious, nostalgic, disappointed, impressed, prepared, jealous, content, devastated, embarrassed, caring, sentimental, apprehensive, faithful

3.1.2 Dataset Selection

We tested multiple data collections to train our models on, and will provide a brief description of the three most promising ones and the reason for our final selection in the following section.

DailyDialog DailyDialog [33], the largest conversational dataset should be certainly considered. It contains more than 13k daily dialogues, and each dialog consists of roughly 8 speaker turns in average. The labels are annotated based on the six basic Ekman emotions. In every dialogue, each utterance is annotated with a label accordingly. Since all dialogues are written by humans, they are less noisy compared to datasets containing posts from social media. Moreover, the conversations involve a wide range of scenarios, such as shopping in a store and school life.

However, as shown in Table 3.2, the distribution of labels is imbalanced. The majority of sentences are labeled with *joy*, while only 74 and 353 sentences are respectively attributed to *fear* and *disgust*. As a consequence, when we train the neural network with DailyDialog, the model cannot learn the characteristics of *fear* and *disgust* sufficiently.

SemEval The SemEval-2018 dataset [40] collects English, Arabic, and Spanish tweets annotated by multiple labels for a competition. It is the largest multi-label dataset and is able to provide us an insight to multi-label situation. The dataset can be split into two parts. The tweets in the first part are labeled by one emotion with its numerous intensity, or the valence value, while the tweets in the second part have multiple labels from 11 given emotion categories. The 11 categories are the six basic Ekman emotions and *anticipation*, *love*, *optimism*, *pessimism*, *trust*. Among them, *joy*, *anger*, *disgust*, *sadness* and *optimism* account for a higher percentage with 39.3, 36.1, 36.6, 29.4 and 31.3 respectively, as shown in Table 3.3. In contrast, *surprise* and *trust* are two of the lowest percentage emotions with only 5.2 and 5.0. Since some emotions are rarely occurred in common conversations, such as *anticipation* and *trust*; some are difficult to detect without context or hashtags, such as *optimism* and *pessimism*, the dataset has to be preprocessed for our task.

Unified dataset As described in the previous part, each dataset has its own advantage and flaws. To improve the overall performance of a dataset, we utilize the unified dataset proposed by Bostan et al. [41]. It aggregates multiple comparably small datasets and merges various schemata into a unified dataset collection. The final unified dataset contains more than 208k sentences from 12 datasets, namely AffectiveText, Blogs, CrowdFlower, DailyDialogs, Electoral-Tweets, EmoInt, Emotion-Stimulus, Grounded-Emotions, ISEAR, SSEC, Tales and TEC.

It consists of 11 labels, which are *anger*, *anticipation*, *confusion*, *disgust*, *fear*, *joy*, *love*, *noemo*, *sadness*, *surprise* and *trust*. They are selected from 59 original labels, where similar labels are combined into new labels. As we focus on six basic emotions, further preprocessing on the unified dataset is also conducted. Specifically *joy*, *love* and *anticipation* are centralized as *joy*; *anticipation*, *confusion*, *trust* and *noemo* are merged into *neutral*.

Dataset	joy	anger	sadness	disgust	fear	surprise
DailyDialogs	74.5%	5.9%	6.6%	2.0%	0.4%	10.6%
Unified Dataset	36.7%	10.9%	19.1%	6.4%	17.0%	9.9%

Table 3.2 Percentage of samples labeled with a given emotion.

Dataset	anger	antic.	disg.	fear	joy	love	optim.	pessi.	sadn.	surp.	trust
SemEval	36.1%	13.9%	36.6%	16.8%	39.3%	12.3%	31.3%	11.6%	29.4%	5.2%	5.0%

Table 3.3 Percentage of tweets labeled with a given emotion.

3.2 Text emotion analysis

3.2.1 Evaluation measures

In single-label classification we can use simple metrics such as

(a) accuracy: $\frac{1}{N}\sum_{i=1}^{N} 1[\hat{y}^{(i)} = y^{(i)}]$

(b) precision: $\frac{t_p}{t_p+f_p}$

(c) recall: $\frac{t_p}{t_p+f_n}$

(d) F-score: $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

with $\hat{y}^{(i)}$ being the prediction for sample $x^{(i)}$ for $i \in [N]$, t_p the number of true positives, f_p the number of false positives, t_n the number of true negatives, and f_n the number of false negatives.

However, in multi-class and multi-label classification, it is more difficult to measure misclassification. A prediction that returns part of the actual labels should be evaluated as more correct than a prediction that does not return any of the actual labels. Therefore, we generalize precision and recall (see Table 3.4): Micro-averaging first sums up all true positives, true negatives, false positives, false negatives for each class and then takes the average; it can be useful when the classes vary in size. Macro-averaging takes the average of precision and recall on different classes; it shows how the system performs overall across the classes. The F-score remains $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$.

	Precision	Recall
Micro	$\frac{\sum_{c \in \mathcal{C}} t_p(c)}{\sum_{c \in \mathcal{C}} t_p(c) + f_p(c)}$	$\frac{\sum_{c \in \mathcal{C}} t_p(c)}{\sum_{c \in \mathcal{C}} t_p(c) + f_n(c)}$
Macro	$\frac{\sum_{c \in \mathcal{C}} \text{Precision}(c)}{\lvert\mathcal{C}\rvert}$	$\frac{\sum_{c \in \mathcal{C}} \text{Recall}(c)}{\lvert\mathcal{C}\rvert}$

Table 3.4 Evaluation measures for multi-class and multi-label classification with $\mathcal{C}$ being the set of classes

3.2.2 Multi-label classification

Multi-label classification assigns to each data sample a set of target labels, i.e. the categorization is not mutually exclusive. As many classification algorithms do not naturally permit the use of more than two classes, we present three approaches to reduce the problem of multi-label classification to multiple binary classification problems: Binary Relevance, Classifier Chain, and Label Powerset.

In Binary Relevance, a single-label binary classifier is trained for each class. Each classifier predicts either the membership or the non-membership of one class. The union of all classes that were predicted is taken as the multi-label output. This approach is popular because it is easy to implement, however it also ignores the possible correlations between class labels.

In Classifier Chain, a chain of binary classifiers $C_0, C_1, ..., C_n$ is constructed, where a classifier C_i uses the predictions of all the classifier C_j , where $j < i$. This way the method can take into account label correlations. The total number of classifiers needed for this approach is equal to the number of classes, but the training of the classifiers is more involved. In particular, the performance depends strongly on the order of the labels.

The Label Powerset takes possible correlations between class labels into account. It is called the label-powerset method because it considers each member of the power set of labels in the training set as a

single label (connecting them by the logical AND function). In the worst-case, this methods needs to train exponentially many classifiers (in the number of classes), i.e. easily leads to combinatorial explosion and computational infeasibility.

3.2.3 Deep learning approaches

LSTM, BiGRU and GRU Models

The appearance and development of deep learning have revolutionized tasks in Natural Language Processing, such as emotion detection. Recurrent Neural Network (RNN) is one of the most well-known models in deep learning. It can take time series into account, and thus is able to process sentences and documents, where words appear in chronological order. Besides, RNN is capable to preserve the contextual information inside conversations with various lengths. In SemEval-2018 task 1 [40], SVM/SVR, LSTMs and Bi-LSTMs become the most widely used algorithms to deal with emotion recognition and classification problem. Therefore, we adopt RNN with LSTM [42], GRU [43] and Bi-GRU to our task, and compare their results in the end.

RNNs consist of a sequence of neurons, which are called hidden states. Each state takes a word from sentence sequentially as part of the input. With directed edges from previous units to current units, each hidden state can also take advantage of previous outputs as part of the inputs, and thus keep the information from previous words. In each time step t, hidden state i combines output from the previous hidden state and input word from the input layer, feeds the concatenation of two parts into the network and obtains output for the current unit i. This process is shown as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>+W_{ax}x^t+b_a}) \tag{3.1}$$

$$y^{<t>} = g2_(W_{ya} + b_y) \tag{3.2}$$

where W_{aa} and W_{ya} denote weight matrices between two hidden states and from input to hidden state respectively, W_{ya} denotes the weight matrix from hidden state to output, b_a and b_y are bias terms respectively.

Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU) are variants of basic RNN, which can avoid gradient vanishing by recording and delivering messages selectively. LSTM is composed of cell, input gate, output gate and forget gate. They control which information from previous states and current input should be memorized, delivered or forgotten. GRU holds similar idea as LSTM, with only reset gate and update gate. Bidirectional LSTM and Bidirectional GRU process input from both forward and backward directions. Therefore, the context of the whole sentence can be preserved and utilized in each hidden state.

In our task, we implement LSTM, GRU and Bi-GRU with similar architecture. The general architecture consists of embedding layer, two RNN layers and a fully-connected layer. The only difference among them occurs in RNN layers. LSTM, GRU and Bi-GRU utilizes two layers of LSTM, GRU and Bi-GRU respectively, as shown in Figure. The output from the last hidden state in the second RNN layer is fed into a fully-connected layer to generate the final result, which is a vector containing seven scores for seven emotions. Each score represents the probability of corresponding emotion for the input sentence. If the dataset is single-labeled, the emotion with the highest probability is selected. In such case, the sum of scores should be 1. In addition, the model can also be exploited for multi-label dataset with modification in the output. Given a manually set threshold, only labels with above-threshold scores will be chosen as the emotions for the sentence.

BERT

Bidirectional Encoder Representations from Transformer (BERT) is a recent model developed by Google AI Language that is widely used in NLP since it can achieve state-of-the-art results in many tasks [44]. BERT

uses a transformer mechanism that is normally constituted of two separate modules: the encoder that processes the text and the decoder that outputs a certain value related to the task in question. However, since BERT's purpose is to generate a language model, only the encoder part is present.

Moreover, BERT pre-trained models, trained on large corpus BooksCorpus (800M words) [45] and English Wikipedia (2,500M words), are available for transfer learning. To build the emotional component of the chatbot, BERT-large-uncased (340M parameters) and BERT-base-uncased (110M parameters), two pre-trained models available in the Hugging-Face transformers library [46], were used as sentence embedding generators, each sentence is transformed into a 1024 one-dimensional vector by BERT-large-uncased and a 768 one-dimensional vector by BERT-base-uncased.

In order to generate the sentence embeddings BERT's first layers use WordPiece tokenization [47] to split the data into tokens and create embedding for every tokens. The embedding for every token are 768 one-dimensional vector BERT-base-uncased and 1024 one-dimensional vector for BERT-large-uncased. A [CLS] token is added in the beginning of the sentence and the [SEP] token at this end of it. The output of the tokenization part is, thus, a matrix of dimension: number of tokens (including [CLS] and [SEP]) by the token embedding vector's dimension.

Then, the tokenization layer's output goes through layers of encoder (transformer blocks), 12 for BERT-base and 24 for BERT-large. Each encoder layer has a self-attention component, multi-head attention in this case, and a feed-forward component. BERT-base uses 12 attention heads and BERT-large uses 16 attention heads in the mutli-head attention component of the encoder. The output of the encoder has the same dimension as the output of the tokenization layer, number of tokens (including [CLS] and [SEP]) by the token embedding vector's dimension.

For sentence classification problems, the [CLS] token's final embedding after running BERT is the sentence's embedding. That one-dimensional of size 1024 (BERT-large) or 768 (BERT-base) is then the input vector of the next layer, a one-layer fully connected network.

When using Unifed Dataset with single label classification, the dimension of the vector after the FC layer is 7. When using SemEval, multi-label classification, the vector's dimension is reduced to 11. After computing the sigmoid function (multi-label classification) or the softmax (single label classification) function on that one-dimensional vector, we could then generate the predictions.

For the multi-label classification problem, since a sentence can be associated to 11 emotions, independently, the prediction vector is thus an 11 dimensional vector where each entry is a probability between 0 and 1 of a certain emotion being present or not in the sentence. Note that the sum of the probabilities do not sum to one; detecting each emotion can be view as an independent binary classification problem. This reasoning justifies the use of the element-wise sigmoid functions on the FC layer's 11 dimensional output vector. Then, a simple average of the binary Cross-Entropy losses applied element-wise on the output vector is used as the network's loss function.

For the single label classification problem, a sentence is associated to one of the seven labels, the basic six Ekman's emotions and *neutral*, the prediction's vector is then a 7-dimensional vector where each entry corresponds to a probability between 0 and 1. It is important to note that this time that, since every sentence is associated to only one emotion, the probabilities sum to 1. This explains the use of the softmax function on the FC 7 dimensional output vector. Then, the multi-class cross entropy loss is applied on the output vector to compute the classification's error.

3.2.4 Classical learning-based approaches

As traditional machine learning methods can outperform deep learning methods, especially when there is not a large amount of training data available, we aim at comparing the results from Section 3.2.3 for

SemEval [40] to a traditional learning approach. The best performing traditional learning methods for emotion detection with respect to accuracy and F-score have been support vector machines (SVM) and naive bayes (NB) classifiers [6, 7, 8]. Thus, in the following we use SVM and NB for emotion classification and compare the results to our deep learning approaches.

SVM computes a hyperplane to separate the data, which has been mapped into some possibly higher dimensional space: we assign a text embedding vector $x \in \mathbb{R}^n$ obtained from FastText (see Section 3.2.3) to the emotion class e if and only if $\beta_e + w_e\phi(x) \geq 0$ for some coefficients $\beta_e \in \mathbb{R}$, $w_e \in \mathbb{R}^m$, and a feature mapping $\phi : \mathbb{R}^n \to \mathbb{R}^m$. An appropriate choice of the feature mapping may allow to handle non-linearly separable data [48]. The hyperplane coefficients β_e, w_e are chosen as a solution to the primal optimization problem of maximizing the distance to a closest data point. Using complementary slackness, it can be seen that only the data points which minimize the distance to the optimal hyperplane, the so-called support vectors, can contribute to the weight vector w_e. Thus, the decision function $\beta_e + w_e\phi(x)$ can be computed as $\beta_e + \sum_{i\in[N]} \alpha_i^e y^{(i)} \langle \phi(x^{(i)}, \phi(x)\rangle$, where $x^{(1)}, ..., x^{(N)}$ are the support vectors and $y^{(i)} : [N] \to \{-1, 1\}$ with $y^{(i)} = 1$ if and only if $x^{(i)}$ is labeled with emotion e. Even though the feature mapping $\phi(x)$ may be very expensive to calculate, the corresponding kernel function $K(x, z) := \langle \phi(x), \phi(z)\rangle$ may be inexpensive. Moreover, as the number N of support vectors is usually much smaller than the number of training data points, SVM is memory efficient and effective in high-dimensional spaces.

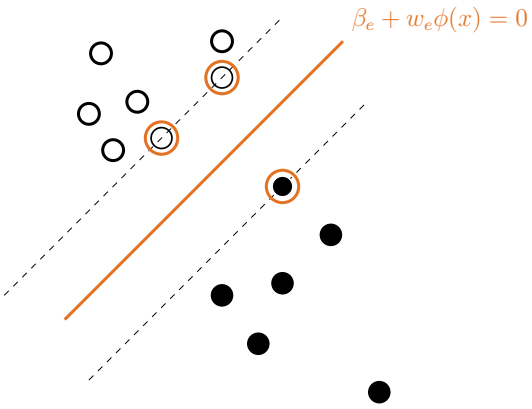


Figure 3.1 SVM: two-class data points (black vs. white), optimal hyperplane (orange line), support vectors (encircled in orange)

NB classifiers use the strong assumption of independence between the features to derive from Bayes' theorem that $p(e \,|\, x) \propto p(e) \prod_{i\in[n]} p(x_i \,|\, e)$ for an emotion e and a text embedding vector $x = (x_1, ..., x_n)$. Then x is assigned to the emotion class that is most probable: $\hat{y} := \text{argmax}_e\, p(e) \prod_{i\in[n]} p(x_i \,|\, e)$. We estimate the class probabilities from the training data. As using FastText for the text embedding yields n-gram features (see Section 3.2.3), we assume the feature distributions to be multinomial [49]. The multinomial NB becomes linear when expressed in log-space [50]. Finally, as it requires only linearly many parameters in the number of variables, it is also highly scalable.

The implementation of single-label binary classifications using SVM and multinomial NB classifier is realized via SVC and MultinomialMB from scikit-learn library. Parameters are tuned using an exhaustive grid search: For SVM, we choose the regularization parameter $C \in \{0.01, 0.1, 1, 10, 100\}$ and the kernel coefficient $\gamma \in \{auto,\ scale\}$, where $auto = \frac{1}{\text{\#features}}$ and $scale = \frac{1}{\text{\#features} \times \text{Var}(x)}$. The regularization parameter is used for a squared l_2-penalty such that low values make the decision surface smooth. Large kernel coefficients determine the influence of each single training example to be strong. For the multinomial NB classifier, we use additive Laplace smoothing parameters $\alpha \in \{1e\text{-}10, 0.1, 1, 5, 10, 100\}$, where 0 represents no smoothing. Priors are chosen from $uniform$ and $learned$. Results for the tuned parameters are presented in Table 3.5. Moreover, results using SVM and multinomial NB in Binary Relevance, Classifier Chain, and Label Powerset are shown in Table 3.6.

Even though good accuracies are achieved compared to state-of-the art deep learning results [6] as well as to our own results using deep learning (Section 3.2.5), precision and recall in the emotion classes "1" show that both SVM and multinomial NB perform poorly in assigning emotion to a SemEval text sample. On the one hand, as the grouping of FastText with respect to semantically similarity might disturb the

emotion classification, another word embedding may achieve better results. On the other hand, the large amount of total available data might allow deep learning to outperform the traditional approaches.

	Support Vector Machine					*Naive Bayes*				
	Accuracy		Precision	Recall	F-score	Accuracy		Precision	Recall	F-score
Joy	0.51	0	0.53	0.77	0.63	0.53	0	0.54	0.78	0.64
		1	0.43	0.20	0.27		1	0.48	0.23	0.31
	$C = 100, \gamma = scale$					$\alpha : 0.0001, uniform\ priors$				
Surprise	0.93	0	0.96	0.97	0.97	0.96	0	0.96	1.00	0.98
		1	0.04	0.03	0.03		1	0.00	0.00	0.00
	$C = 100, \gamma = auto$					$\alpha : 1e\text{-}10, learned\ class\ priors$				
Neutral	0.89	0	0.96	0.93	0.94	0.96	0	0.96	1.00	0.98
		1	0.09	0.15	0.11		1	0.00	0.00	0.00
	$C = 100, \gamma = scale$					$\alpha : 0.0001, learned\ class\ priors$				
Sadness	0.66	0	0.70	0.89	0.78	0.48	0	0.72	0.41	0.52
		1	0.31	0.11	0.17		1	0.31	0.63	0.42
	$C = 100, \gamma = scale$					$\alpha : 1e\text{-}10, uniform\ priors$				
Anger	0.60	0	0.65	0.84	0.73	0.52	0	0.66	0.54	0.59
		1	0.36	0.16	0.22		1	0.37	0.49	0.42
	$C = 1, \gamma = auto$					$\alpha : 1e\text{-}10, uniform\ priors$				
Fear	0.79	0	0.87	0.89	0.88	0.86	0	0.86	1.00	0.93
		1	0.21	0.19	0.20		1	0.00	0.00	0.00
	$C = 100, \gamma = scale$					$\alpha : 1e\text{-}10, learned\ class\ priors$				
Disgust	0.61	0	0.65	0.88	0.75	0.52	0	0.65	0.57	0.60
		1	0.40	0.14	0.20		1	0.37	0.45	0.41
	$C = 1, \gamma = scale$					$\alpha : 1e\text{-}10, uniform\ priors$				

Table 3.5 Results and tuned parameters of a single-label binary classification on the SemEval dataset using SVM and multinomial NB classifier. We use SVC and MultinomialNB from scikit-learn library. "1" describes the class having the respective emotion label, "0" the class which does not have the respective emotion label. Running times for training the classifiers with fixed parameters are negligible.

	Micro-scores				Macro-scores				Hamming loss	Subset accuracy
	Precision	Recall	F-score	Jaccard	Precision	Recall	F-score	Jaccard		
SVM	0.32	0.46	0.38	0.23	0.28	0.37	0.27	0.17	0.37	0.09
NB	0.46	0.27	0.34	0.21	0.1	0.15	0.1	0.1	0.25	0.32

Table 3.6 Evaluation of Binary Relevance, Classifier Chain, and Label Powerset using SVM and multinomial NB classifier on the SemEval dataset. Only the results for the respectively best method are presented, i.e. Classifier Chain for SVM and Label Powerset for NB.

3.2.5 Deep learning models comparison

Various models were used to train the emotional component of the chatbot, we compare the training results in Table 3.7. The models were first trained on the SemEval dataset, a multi-labels dataset where every sentence can be associated with more than one emotion (11 labels per sentence). A threshold of 0.8 (and also 0.5 for BERT) determines if a certain emotion is predicted for a given sentence, the probability asso-

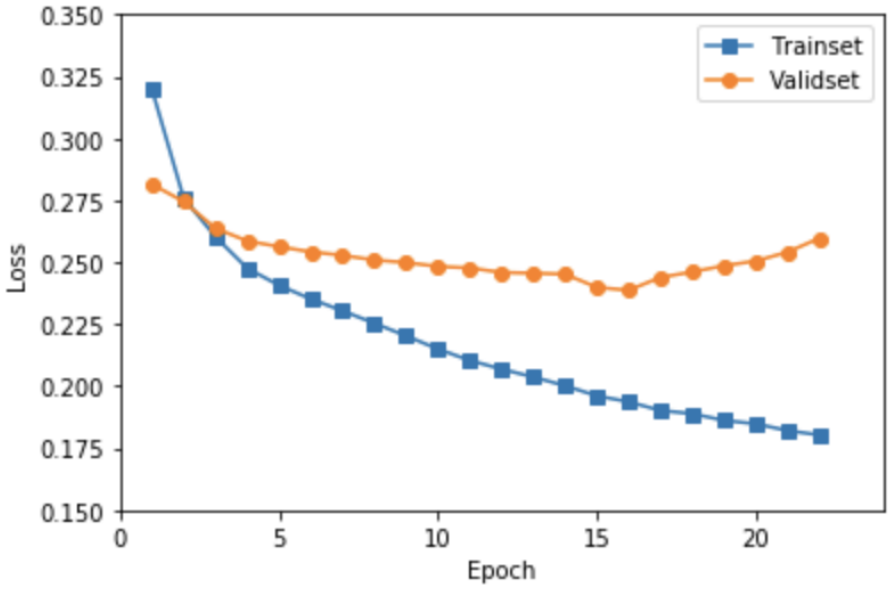


Figure 3.2 Loss in training period of RNN with LSTM

ciated with it in the final output vector has to exceed the threshold. Then, the models were also trained on the Unified Dataset, a single-label dataset, where one sentence is associated with only one emotion. This time, the probabilities on the final output vector are summing to one; the model's prediction corresponds to the emotion with the highest probability.

Three metrics, namely precision, recall, and F-score are computed for each six basic emotion labels. The macro-average of those metrics is also computed for each model. All the final results are shown in Table 3.7. We used the validation loss and performed early stopping to prevent the over-fitting phenomenon. For example, Figure 3.3 shows the change of training loss and validation loss for RNN with LSTM model on Unified Dataset. The curve indicates that the model after 16 epochs is preferred because the validation loss starts to increase after that point.

We see clearly that BERT models, BERT-large-uncased with a threshold of 0.8 and 0.5, achieve better results on SemEval, reaching better macro-averaged metrics. This result indicates that using a more complex model is beneficial for our task since it can capture more complex information in the sentences that may not be captured by a more simple structure. To see the threshold's effect on the results, we compared the performance of BERT, changing only the threshold from 0.8 to 0.5. The BERT used model is identical in both cases: the same weights that we obtained after training the model are used for BERT(0.8) and BERT(0.5). BERT(0.8) achieved high precision for the different labels, (over than 0.85 for 5 of the 6 basic emotions except for *surprise* for which we obtained 0.5. BERT(0.5) has lower precision for the labels, however, it reaches higher recall for emotions and the highest macro-averaged F-score, making it the best model. The macro-averaged score is preferred to compare the models since we value both precision and recall.

Joy seems to be the label for which we obtained better F-scores as it is the emotion that is more present in the dataset (39,2% of the tweets are labeled with that emotion). On the opposite, the F-scores are the lowest for *surprise* and it could be explained partially by the SemEval class distribution as only 5.2% of the tweets contain *surprise*.

After training the models on Unified dataset, we achieved better results for the three RNN models. It shows the advantages of a larger and a more balanced dataset. Due to the absence of powerful computers

available, we could not train the BERT model on the Unified Dataset which has significantly more data sentences. However, BERT trained on the SemEval still leads to the best results among all models, since BERT has a powerful structure to capture a more complex data structure.

Model(Thres.) Dataset	Metrics	Anger	Disgust	Fear	Joy	Sadness	Surprise	Macro Average
RNN-LSTM(0.8)	Precision	0.53	0.50	0.52	0.71	0.70	0.40	0.560
	Recall	0.47	0.55	0.45	0.60	0.57	0.27	0.485
SemEval	F-score	0.498	0.524	0.482	0.650	0.628	0.323	0.520
RNN-BiGRU(0.8)	Precision	0.50	0.46	0.48	0.73	0.55	0.42	0.523
	Recall	0.34	0.34	0.35	0.66	0.56	0.32	0.428
SemEval	F-score	0.405	0.391	0.405	0.693	0.555	0.363	0.469
RNN-GRU(0.8)	Precision	0.34	0.26	0.38	0.64	0.54	0.37	0.422
	Recall	0.29	0.23	0.37	0.41	0.65	0.34	0.385
SemEval	F-score	0.313	0.244	0.375	0.523	0.590	0.354	0.423
BERT(0.8)	Precision	0.889	0.858	0.897	0.943	0.925	0.5	0.83
	Recall	0.635	0.571	0.579	0.703	0.419	0.393	0.54
SemEval	F-score	0.741	0.686	0.704	0.805	0.577	0.440	0.65
BERT(0.5)	Precision	0,815	0.782	0.754	0.883	0.763	0.508	0.74
	Recall	0.781	0.799	0.736	0.795	0.608	0.411	0.69
SemEval	F-score	0.797	0.791	0.745	0.837	0.676	0.454	0.71
RNN-LSTM	Precision	0.701	0.700	0.749	0.910	0.854	0.533	0.741
	Recall	0.476	0.472	0.449	0.532	0.601	0.471	0.500
Unified	F-score	0.567	0.564	0.546	0.671	0.706	0.500	0.597
RNN-BiGRU	Precision	0.700	0.691	0.692	0.859	0.822	0.492	0.709
	Recall	0.432	0.402	0.452	0.5	0.528	0.453	0.461
Unified	F-score	0.534	0.508	0.547	0.632	0.643	0.472	0.558
RNN-GRU	Precision	0.690	0.703	0.730	0.906	0.845	0.542	0.736
	Recall	0.380	0.374	0.423	0.441	0.477	0.403	0.416
Unified	F-score	0.490	0.488	0.536	0.593	0.610	0.462	0.532

Table 3.7 Results for models with SemEval and Unified Dataset.

Dataset	Model (Thres.)	Accuracy
SemEval	LSTM (0.8)	66.87%
	Bi-GRU (0.8)	64.02%
	GRU (0.8)	59.28%
	BERT (0.8)	86.84%
	BERT (0.5)	88.24%
Unified	LSTM	85.89%
	Bi-GRU	81.93%
	GRU	80.67%

Table 3.8 Accuracy comparison for models with SemEval and Unified Dataset

In our task, we should consider both performance and computational time equally. Although RNN models are not as accurate compared with BERT models, their training time is much lower and the response time during the actual conversation is significantly lower. Therefore, we consider choosing the RNN-LSTM

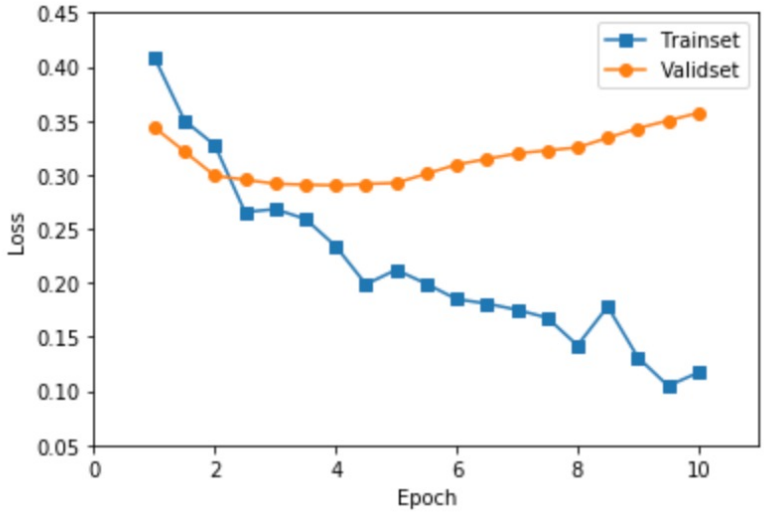


Figure 3.3 Loss in training period of BERT-large model trained on SemEval dataset

for our model, which has higher precision and F-score, and also accuracy among the three models (Table 3.8). In Section 3.3.3, we also test our models with a 60-sentences dataset, and the result also supports our point of view. In the future, it is possible to exploit lightweight BERT, such as ALBERT, or preload BERT to decrease the time cost and preserve the performance at the same time.

For reproducibility, we would like to introduce the hyperparameters for our models: For three RNN models trained on SemEval and Unified Dataset, the embedding dimension is 100, the learning rate is set as 5e-4, the hidden state's dimensions in the RNN layers are 64 and 32 respectively, and the dropout is set as 0.5 and 0.4 respectively. For BERT-large, fine-tuned on SemEval, we used a learning rate of 2e-05, a batch-size of 64, weight-decay 3e-06 and we trained the model for 10 epochs. We chose the model's weights after 3 epochs since after then, the validation loss increases.

3.3 Hierarchical emotion detection

As seen in the previous section, the pre-trained models enable us to predict the emotion of the user based on a text analysis. To have a more robust emotion estimation, several preprocessing steps are performed on the user message. In addition, the emotion expressed using emojis and emoticons is merged with the text emotion to predict an overall emotion.

3.3.1 Emoji and Emoticon

In today's online communication, emojis and emoticons are becoming one of the main modern languages that allow fast, diverse and above all universal communication. An emoji (exp: 😄😉) is "a small digital image or icon used to express an idea or emotion. The word emoji essentially means "picture-character" (from Japanese e - picture, and moji - letter, character)" [51]. Whereas an emoticon (:) :-]) is "a representation of a human facial expression using only keyboard characters such as letters, numbers, and punctuation marks" [51]. Emojis and Emoticons facilitate human emotional expressions by representing their facial expression (smileys, tears, etc.) using some representations and therefore provide valuable information, particularly in sentiment and emotion analysis [52]. Another potential in emojis is that they are language-independent indicators of emotions, which can help to avoid errors of language processing [53].

To extract emojis and emoticons from the text, a database of some of the most frequent emojis and emoticons as well as the emotions they express is created and maps emojis / emoticons to the VAD space as well as the six basic Ekman emotions. The most expressed emotion is recorded as an emoji / emoticon

emotion and will then be combined with the emotion of the text. In the case that no emoji or emoticon is given in the user message, the neutral emotion will be selected.

3.3.2 Sentiment Analysis

Sentiment Analysis is a text analysis method that focuses on detecting the sentimental polarity (positive, negative, or neutral) within a text, based on the opinions and emotions expressed in the text [54]. In the investigated hierarchical emotion detection model, the sentiment analysis component can be considered as a classification process to detect the neutral user message frequently used in conversations with chatbots (e.g. greetings, order confirmation, etc.). In fact, once a user message has neutral polarity, the neutral emotion will be labeled without using the pre-trained text prediction models. A negative or positive polarity indicates that the user message has an emotion and therefore a prediction using the emotion extraction models is necessary. By using the sentiment component, the complexity and the computational effort of the algorithm is reduced, and the reliability of the neutral emotion is increased.

3.3.3 Key-word approach

Recognizing emotions in text is an important task of natural language processing (NLP). This task is particularly challenging when the emotion is hidden implicitly in the text, and therefore, its solution requires an understanding of the context [6]. Nevertheless, not all thoughts and emotions require understanding of the complete sentence or its context. Certain explicit emotions would be directly recognized if certain words are present in the user message.

Keyword-based approaches for explicit emotion recognition have been investigated [55]. For instance, the sentence “I am happy with your service” explicitly expresses *joy* and includes the emotion keyword “happy”. A keyword-based approach would be able to recognize emotion successfully. However, the presence of an emotion keyword does not always correspond to the emotion expressed. For example, the phrases "I am not happy at all" include the emotion keyword “happy" but do not express that emotion.

To solve this problem, a negation check is carried out after each emotion keyword detection in which we check if a negation word ("no", "not", "don’t", etc.) is present in the sentence and linked to the keyword. In case of negation, the opposite emotion will be predicted. For example in the sentence "The drink is not so good!" the keyword "good" will be detected which belongs to the emotion *joy*, but after the negation check (the presence of "no"), the emotion *sadness* will be predicted. To avoid the problem of detecting a negation word not related to the emotion expressed by the user, we only considered negation words located at most 3 words before the explicit emotional keywords.

To validate the keyword approach, we compared the results of the emotion extraction models trained on the SemEval dataset with and without the keyword approach and using a threshold of 0.8. Table 3.9 shows the percentages of correctly detected emotions of 60 emotional sentences (10 sentences for each of the 6 basic emotions) and proves that the keyword approach improves the prediction of investigated deep learning models.

Model	GRU	BGRU	LSTM	BERT
Without key-word	12.07%	13.79%	36.21%	58.62%
With key-word	62.79%	62.07%	65.52%	70.69%

Table 3.9 Comparison of the accuracy of models with and without the keyword approach

3.3.4 Fusion of text and Emoji-Emoticon detections

After extracting the emotions from the text, emojis and the emoticons, a merging process is performed to predict the global emotion of the user. A map containing all possible combinations of categorical emotions allows the fusion and enables to predict complex emotions such as *sarcasm*. For example, for a user’s

message "Please bring me a cola :) to the pool." the emotion of the text would be neutral, and the emoticon emotion would be *joy*. By combining the two extracted emotions, the global emotion *joy* will be predicted. *Sarcasm* will be predicted for example in case of a positive text emotion (*joy* or *surprise*) combined with an emoji or an emoticon expressing *anger* or *disgust*. The complete combination mapping is presented in the appendix. Moreover, in case of an unclear emotion's combination such as contradiction of *joy* and *sadness*, the fused emotion is labeled as confusion. Once a confusion is detected, the bot should ask the user to repeat his request.

Figure 3.4 summarizes the complete hierarchical emotion extraction model and how the global user emotion is predicted. This fused global emotion is then used as a main entity in the Rasa-core to create appropriate stories that allow the bot to learn to react to the different emotions of the users.

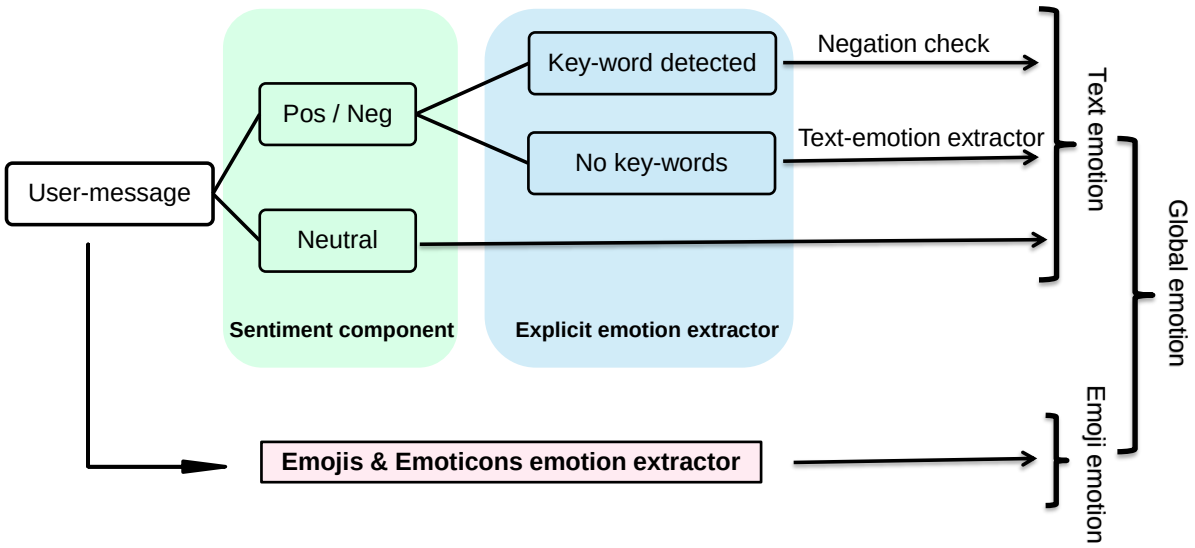


Figure 3.4 Hierarchical emotion detection model

4 Chatbot Integration

As explained in the Introduction, we implemented our project as a Rasa Chatbot coded in Python3 that can be chatted with locally through the terminal, or online through for example the integrated website interface RasaX or Telegram.

The Rasa framework can be split into two parts, the NLU, a processing tool for intent classification, response retrieval and entity extraction and the dialogue engine Rasa Core. For a proof of concept the implemented bot works in a "Bartender Domain". We were provided with a fitting NLU data set including names of drinks, places and typical customer intents (i.e. to order drinks).

4.1 Rasa NLU

To understand how we integrated some of the more complex parts, it is helpful to start with the basic concepts of Rasa NLU. By default, a NLU training data file is required, that contains example sentences with the message intent and entities [56]. In our case for example the intent could be to order drinks, and mentioned entities are a temperature and a drink name such as Cosmopolitan. During a conversation entities can be saved in so called slots, and their current value is tracked and can be retrieved by the bot [57]. All possible intents, entities and slot-mappings have to be listed in the domain file, as well as the custom bot actions which will be explained in the Rasa Core Section 4.2, as well as a more detailed overview of the component interactions in Figure 4.4.

4.1.1 Emotion component

As mentioned in Section 3.3, each message is used to predict the user's emotion and then generate an appropriate response. The complete hierarchical emotion detection is implemented in Rasa as a custom NLU Component. A custom component is a message processing unit in a pipeline used to perform a specific task which NLU does not offer (for example, sentiment analysis, emotion analysis, etc.). Components are collected sequentially in a pipeline. Each component is called one after another. This holds for initialization, training, persisting, and loading the components. During each of these steps, components can pass information to other components [58].

The emotion component reads first a configuration file where the text emotion analysis model (LSTM, BiGRU, BERT, etc.) is specified. Second, the emoji and emoticon as well as the key words databases are loaded as data frames. Finally, the global emotion is predicted following the hierarchical model (Figure 3.4). The predicted global emotion is then returned as an entity "emotion" and could be tracked in Rasa-actions as a slot.

4.1.2 Pipeline optimization

Chatbots technology relies greatly on the capacity of correctly identifying through the text the intents, user's intentions, and the entities, words that express valuable information.

This intent/entity identification is performed after every user's message during the conversation. Indeed, after receiving the user's message, the bot tries to correctly classify that message into predefined categories of intents while simultaneously extracting entities.

In our case, there are 15 different intents: *Greet, goodbye, bot-challenge, confirm-negative, confirm-positive, cancel-order, how-are-you, ask-for-options, order-drinks, recommend, thanks, inform, feedback,*

mood-great, mood-unhappy. Some of them are specific to our domain (order-drinks, ask-for-options, cancel order, etc.), while others are more general and can be present in a wide range of conversations. (greet, goodbye, thanks, etc.)

In addition to the intents, there are six possible types of entities: *AMOUNT, LOCATION, SIZE, DRINK, TEMPERATURE, personality, emotion, mood*. The capitalized ones are related to key elements of the user's order and the others to the bot's state: its personality, its mood, and the emotion it has detected.

The relevance of the bot's responses relies greatly on those two classification problems: intent and entity classification. Rasa has built-in tools to perform those tasks. The pipeline defining the text pre-processing steps and the intent/entity classifier has to be indicated in a configuration file.

We compared three different pipelines frequently used by Rasa users and compared their results on the bartender dataset (nlu dataset file). All three use Dual Intent and Entity Transformer Classifier (DIETClassifier) for intent/entity classification, but use different tokenizer and featurizer during the pre-processing steps. Figure 4.1 shows the three different pipelines.

```yaml
# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: "char_wb"
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
  - name: MemoizationPolicy
  - name: FormPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
  - name: MappingPolicy
```

(a) config

```yaml
# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en
pipeline:
  - name: ConveRTTokenizer
  - name: ConveRTFeaturizer
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
  - name: MemoizationPolicy
  - name: FormPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
  - name: MappingPolicy
```

(b) config-2

```yaml
# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en
pipeline:
  - name: HFTransformersNLP
    model_weights: "bert-base-uncased"
    model_name: "bert"
  - name: LanguageModelTokenizer
  - name: LanguageModelFeaturizer
  - name: DIETClassifier
    epochs: 100
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
  - name: MemoizationPolicy
  - name: FormPolicy
  - name: TEDPolicy
    max_history: 5
    epochs: 100
  - name: MappingPolicy
```

(c) config-3

Figure 4.1 pipeline configuration files

We used DIETClassifier since it showed state-of-the-art results for entity and intent classification in a conversational setup [59]. We trained it on our given bartender dataset. It can be used with trained (config) or pre-trained (config2 and config3) tokenizers and featurizers.

The first configuration uses the WhiteSpaceTokenizer pre-defined in Rasa that creates tokens by simply detecting whitespaces between words: the sequence of characters between two spaces is defined as the token. To create the token embeddings, that pipeline uses two different featurizers that are trained from scratch with the bartender nlu dataset: RegexFeaturizer, LexicalSyntacticFeaturizer, and CountVectorFeaturizer.

The second pipeline uses ConveRT (Conversational Representations from Transformers) [60], a pre-trained language model on Reddit data that is trained for response selection tasks. The word pre-trained representations from the encoder are widely used in conversational settings as word embeddings. Indeed, it is currently the model that Rasa recommends to its users building a chatbot in English.

The third pipeline is using BERT-base-uncased pre-trained model to generate word embeddings. BERT, which we introduced earlier, is used in multiple Natural Language tasks as it offers often better results [45]. However, its number of parameters and the fact that it requires high computational capacities during training and test time could outweigh its potential in terms of performance.

We trained the three pipelines with our bartender dataset where intents and entities are defined. Each pipeline was trained 5 times, each time by excluding a certain number of the training data (lines in the NLU training data file): 0%, 25%, 50%, 70%, and 90% respectively. The purpose is to determine how the pipeline performs on our domain's text distribution.

The label-weighted average F-score (F-score) for intents and entities on the test set for each exclusion percentage was then recorded after training. The results are presented in the following graphs. The weighted F-scores for intents and entities are computed on the test set. The weighted F-score over intents and entities is a new metric that we used in order to compare the three different pipelines.

For intents it is defined as: $F\text{-}score_{weighted} = \sum_{i \in intents} w_i * F - score_i$, where w_i corresponds to the number of occurrences for intent i over the total number of intents and $F - score_i$ the F-score on intent i. For entities it is defined as: $F\text{-}score_{weighted} = \sum_{e \in entities} w_e * F - score_e$. where w_e corresponds to the number of occurrences for entity e over the total number of entity and $F - score_e$ the F-score on entity e.

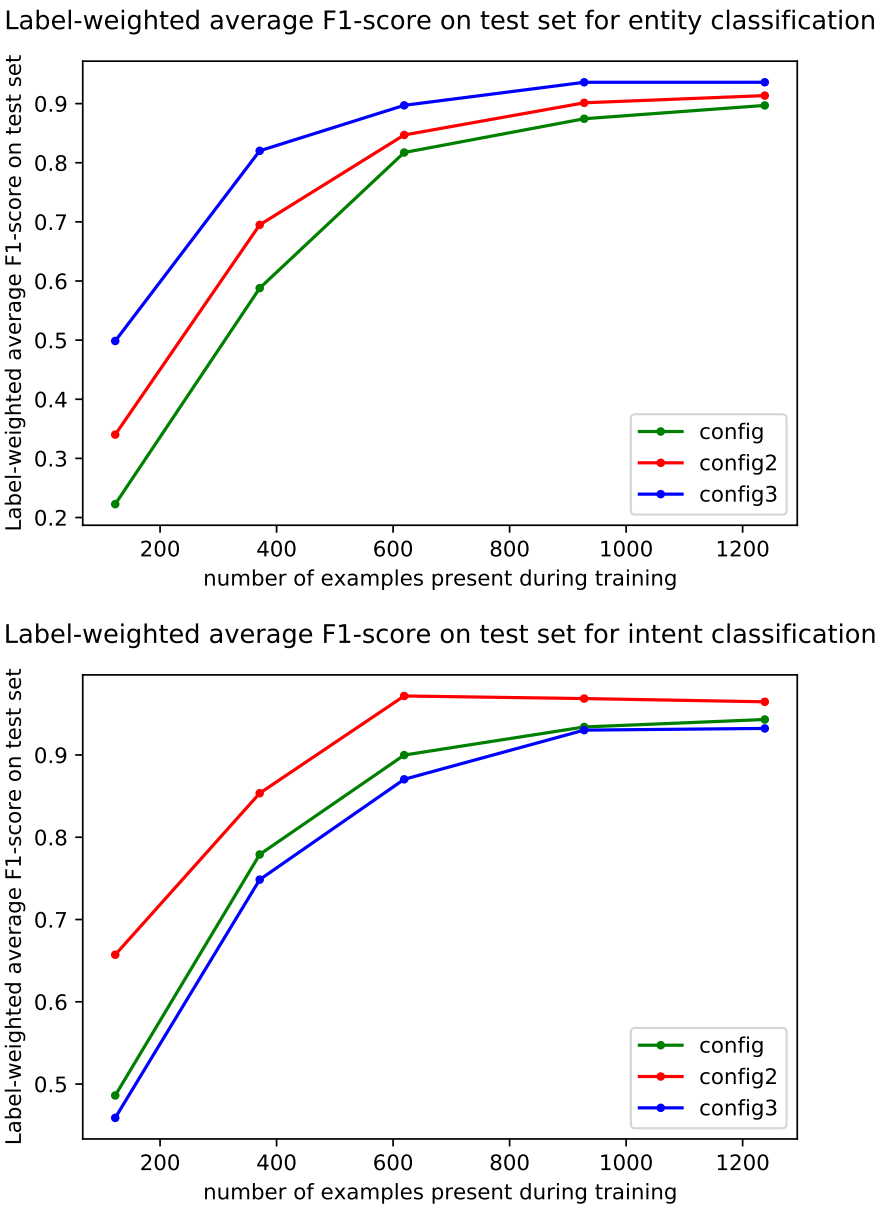


Figure 4.2 Pipeline comparison for entity and intent classification

We clearly see that config-2 performs better for intent classification while config-3 performs better for entity classification. Since every user's message is associated to one intent by the bot in order to select the correct response and that some user's messages do not contain any entity (greeting messages, positive/negative confirmation, goodbye, etc.), the intent classification remains a more important task and a

pipeline achieving higher performance should be prioritized. Moreover, config-3, involving BERT requires more computational power and, thus, requires powerful machines equipped with GPUs in order to perform the task in a reasonable time. Thus, config-2, a lighter pipeline, is preferred.

4.2 Rasa Core

The Core is the Dialogue Engine of Rasa that operates based on the defined stories, domain, actions, slots and forms [61]. As already mentioned, the domainfile contains all possible intents, entities, responses, actions and entity slots. An overview is given in Figure 4.4, where the interactions between the components are shown. The NLU file, how the user message gets processed and intents and entities are detected has been talked about before and is not directly a Rasa Core part. The stories on the other hand, are conversational training data used to train the Rasa dialogue management models [62].

A story is a represents a conversation between a user and the AI assistant and is written in a specific format where user messages are stylized with only their intents (and entities where necessary) while the responses of the assistant are expressed as the corresponding action names. For many conversational situations there are examples in the file, which suggest the bot which actions to execute, depending on entities and intents that are present.

```
## happy path empathetic
* greet
  - update_mood
  - reply_greet
* how_are_you
  - update_mood
  - reply_how_are_you
* mood_great{"PERSONALITY":"empathetic"}
  - update_mood
  - utter_happy
```

Figure 4.3 Rasa Stories Example

Figure 4.3 is an example of a story with the name *happy path empathetic*, where user message intents are marked by a star "*" and bot responses by a dash "-". Marked in red are the entity slots, in this case the bot personality is the decisive factor. In a conversation scenario where the user expresses that he is in a great mood. Here only the empathetic bot will respond with *utter_happy*, the apathetic with something different defined in a similar story. The bot can also be calibration further after setting the story and NLU data, with the option to use the interactive learning interface Rasa X. Here the user can directly correct false classifications, actions and save successful conversations.

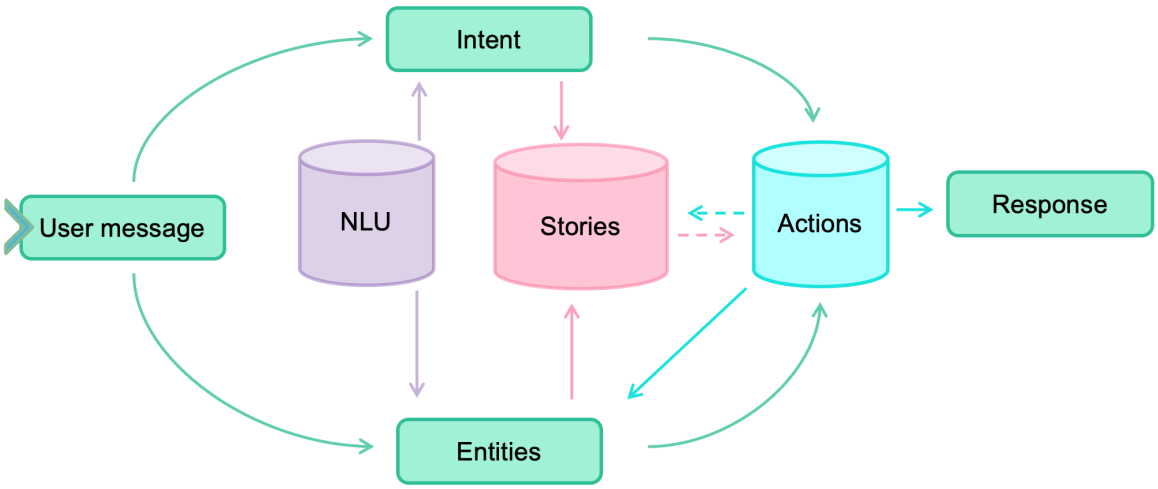


Figure 4.4 Rasa Component Interaction

4.2.1 Custom actions

As explained in the section before, the Rasa Core trains on the stories file, containing stylized conversation examples, but it is not designated to change the response style depending on user or bot emotions directly. For this special task, as well as for some service specific actions, self made custom actions have to be implemented and included in the stories. These custom actions can also write and change entity slots as bot mood for example, which will be explained in detail in the following chapter. A also very helpful tool for customer service conversations is the integrated class FormAction, that can be used to make sure a list of entities was given before moving on.

4.2.2 Personalities and moods

The bot at this stage is equipped with an intent/entity classifier component, an emotional component that can detect the user's emotions, and a response selector trained with the rasa stories file (conversations skeletons) to respond correctly to the user's requests. To mimic human interactions, the bot should not remain stoic and should also express certain emotionality and have different moods.

The present bot is able to achieve that goal without mirroring the user's emotions. Indeed, the bot created is not simply responding and expressing the last emotion detected in the user's message. A bot doing so would constantly change its style of responses and disrupt the natural flow of the conversation. However, in order to differentiate that bot from a generic bot, it has to remain sensitive to the user's emotions and adapt from time to time its style of responses. Thus, the user expressing *anger* for the first time should not trigger a change of style of response. However, the user expressing *sadness* repeatedly should definitively lead to a certain change. The following formula presented in Section 2.3 controls the update of the bot's mood:

$$\text{mood}_{\text{new}} = \text{mood}_{\text{current}} + 0.90\,(\textit{neuroticism} + 1)\,\text{emotion}.$$

We have found through empirical testing that a value of 0.9 leads to better results. The bot's mood is reactive to customers' emotions, without being too volatile. Indeed, the bot's mood changes approximately after 2 or 3 repeated emotions expressed by the user.

As explained in Section 2.2, in order to be used in different business contexts, the present bot possesses two different personalities: empathetic and apathetic (or business). The user, at the beginning of the conversation, can define the personality of the bot by simply typing *I would like to talk to an empathetic bot*, for example.

The selected personality will then influence the emotional reactivity of the bot: the empathetic is sensitive to the user's emotions, while the apathetic bot remains always in the neutral mood. As defined in Section 2.3, the bot has three different moods: *friendly*, *neutral*, and *mean*. The response style is completely defined by the bot's mood. However, it is constantly changing in the course of the conversation. When in a *friendly* mood, the bot uses a friendly tone (emoticons and positively connoted words). The *neutral* mood imposes an informative and sober tone. Finally, when in the mean mood, the bot is irritated by the user and the responses are short and contain negatively connoted words.

For every type of response, there are three different choices of utterances, one for each mood. To send the menu, for example, the chatbot can select depending on its mood from the three following answers:

(a) Friendly: Here's a list of our drinks 😁. Please tell me when you're done or need some advice :) .

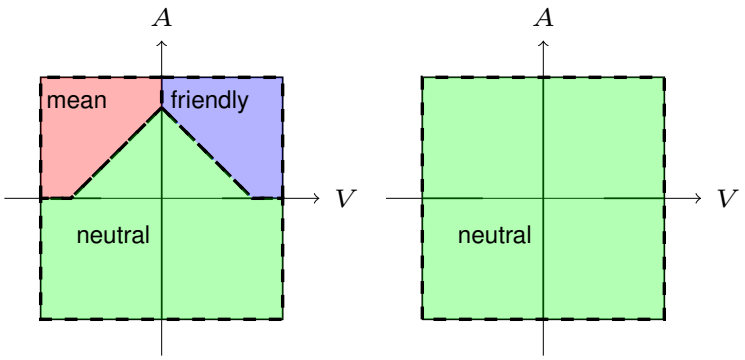
(b) Neutral: Here's a list of our drinks. Please tell me when you're done or need some advice.

(c) Mean: Here's a list of our drinks. Tell me what you want.

The mood can be associated with a point in the *VAD* space ($[-1, 1]^n$) introduced in section 2 and is updated after every user's message. The categorical representation is also updated with respect to the updated *VAD* representation of the mood.

Dominance refers to the control over emotion that one can have, the negative values expressing a greater control. Thus, if the mood's dominance value is negative, the mood is automatically categorized as *neutral*. Indeed, the bot can have control over its emotions and then chooses to express neutral responses. However, when the mood's dominance is positive, the bot loses that sense of control and can be *friendly* or *mean*. For that case, we defined three different regions in the Valence-Arousal space.

Furthermore, in order to redirect unsatisfied customers to real customer service agents, we introduced a function counting the number of times a customer repeatedly expresses negative emotions (anger, sadness, or disgust). If the negative emotions sequence's length exceeds 5, the user is automatically redirected to a real bartender.



(a) dominance > 0 **(b)** dominance ≤ 0

Figure 4.5 Mood's categorical representation according to the mood's *VAD* representation for the empathetic bot's personality. If the mood dominance value is > 0, then $H^{\geq}_{((1,1),\beta)}$ and $H^{\geq}_{((-1,1),\beta)}$ define the zones, where $\beta = 0.75$. Otherwise, the mood is equal to *neutral*.

4.3 Sample conversations

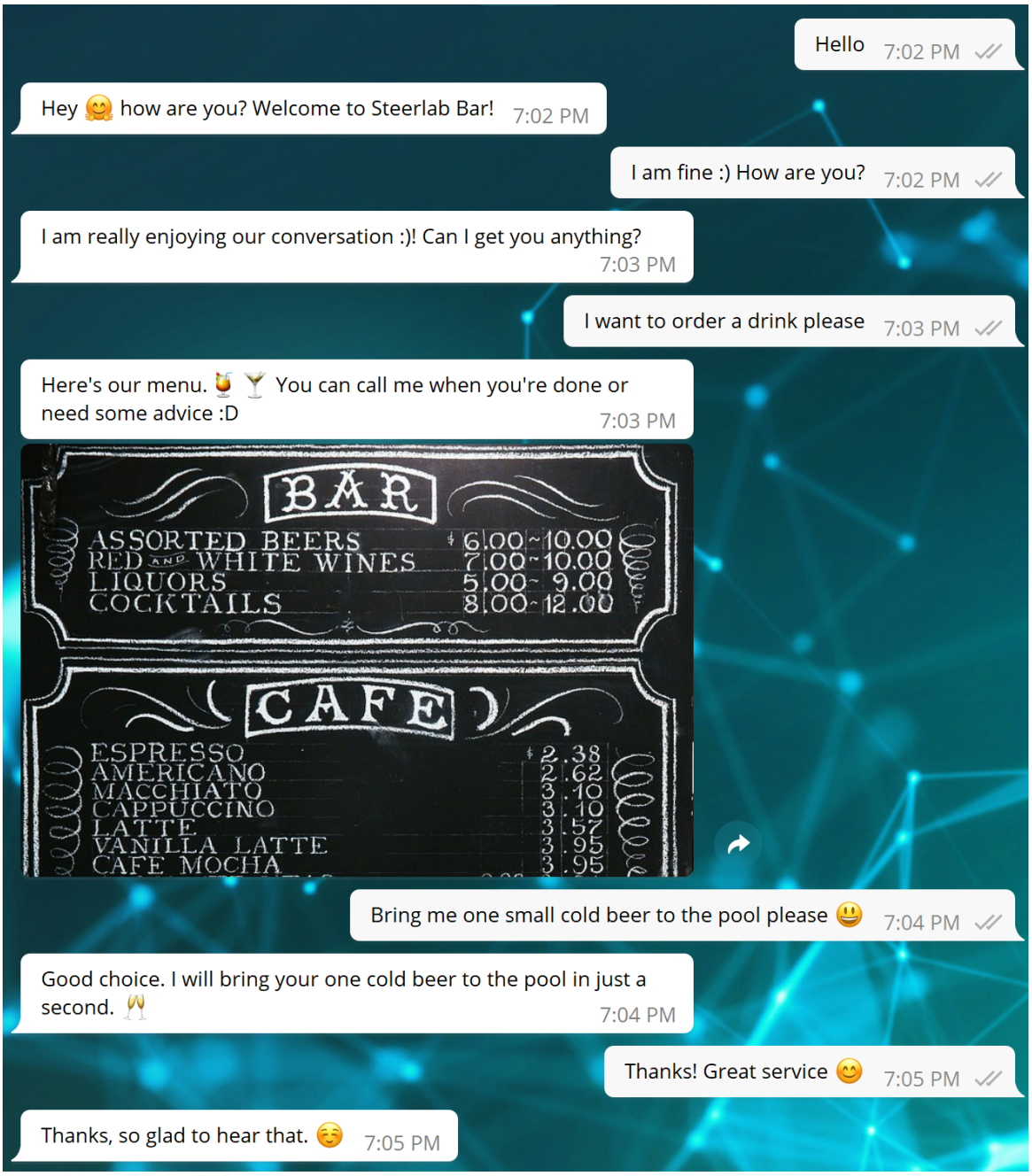


Figure 4.6 Telegram Sample Conversation

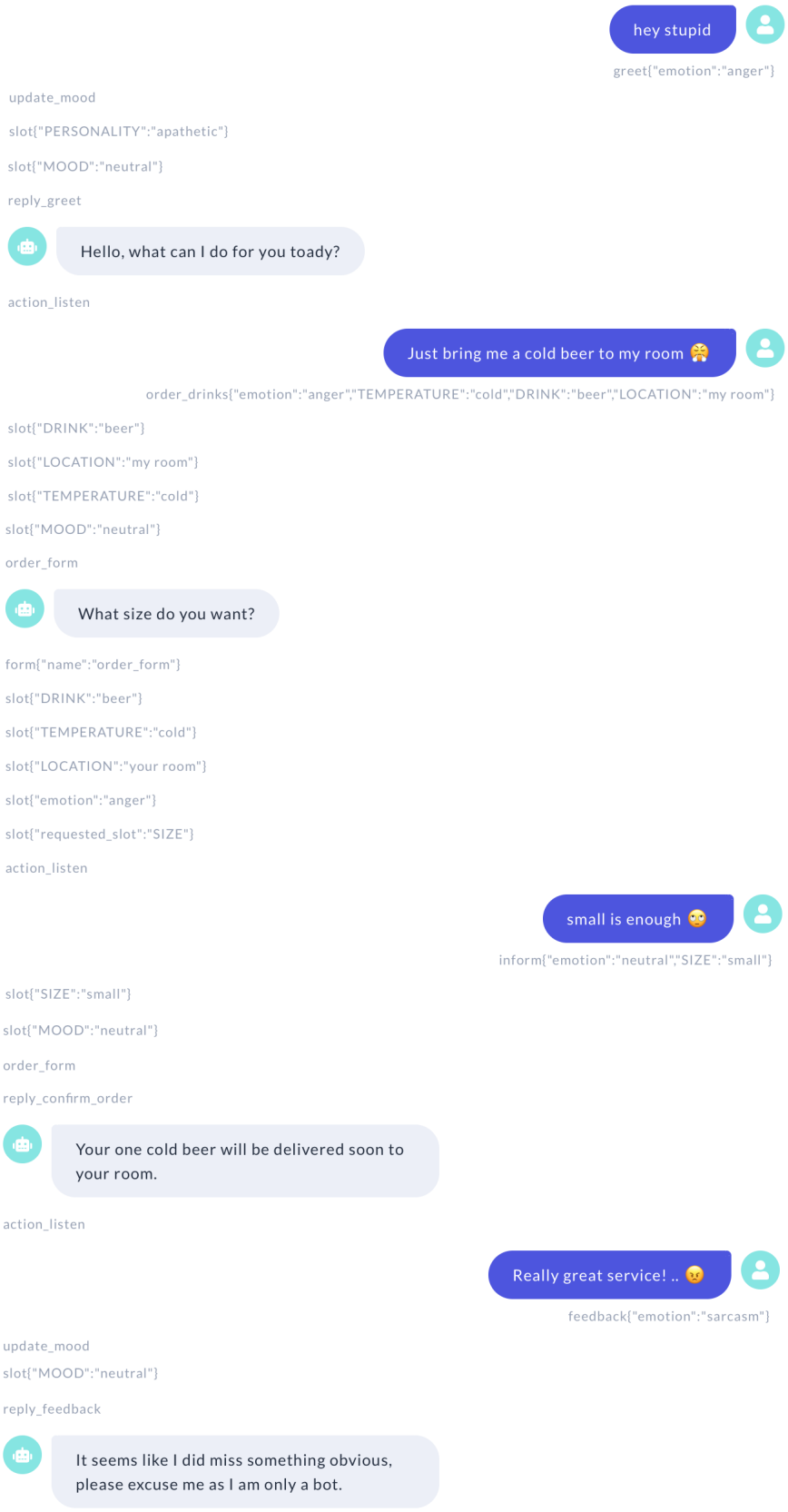


Figure 4.7 Rasa X Sample Conversation

5 Conclusion

The project goal was to develop an affective personal assistant with the ability to have its own sentiments and emotional reactions. We achieved this goal and designed an emotional bot using the Rasa framework. Using a hierarchical emotion extraction model, the bot successfully detects the six basic user emotions, forms its own mood and reacts to the detected emotions based on its personality.

The accuracies of our emotion detection can compete with human accuracies, which can never attain 100% [9]. Given the fact that every individual perceives emotion differently, it is not possible to classify all emotions correctly with only text input, without knowing the full background of the texting person. In this project, we relied on chosen scientific psychological concepts for modeling emotions and we designed the emotion extraction model such that it can be easily adapted to different psychological approaches and different platforms.

The emotional chatbot has been tested in different conversations where users express different emotion patterns and has shown satisfiable results with both a friendly and a business oriented personality. Other personalities could be easily added in the future, as well as a more excessive action, NLU or stories file. With changes to these files the bot can also adapt to other domains. Other potential extensions include connecting the system to a speech and image recognition models, which could increase the reliability of emotion recognition, but would come with a great deal of implementational work. Aside from that, expanding the emotional capabilities of the chatbot would be an interesting extension, such as giving the bot morals, his own values or further character traits.

Working on the Rasa platform and implementing an emotional chatbot was a very enriching experience for all of us and we would like to thank Horváth & Partners and the mentors for their support as well as for offering this group project.

A Supplementary material

Text emotion	Emoji emotion	Global emotion	Text emotion	Emoji emotion	Global emotion
joy	*joy*	*joy*	*disgust*	*fear*	*disgust*
joy	*surprise*	*joy*	*disgust*	*anger*	*anger*
joy	*disgust*	sarcasm	*disgust*	*neutral*	*disgust*
joy	*sadness*	confusion	*fear*	*joy*	confusion
joy	*fear*	confusion	*fear*	*surprise*	*joy*
joy	*anger*	sarcasm	*fear*	*disgust*	*disgust*
joy	*neutral*	*joy*	*fear*	*sadness*	*sadness*
surprise	*joy*	*joy*	*fear*	*fear*	*fear*
surprise	*surprise*	*surprise*	*fear*	*anger*	*anger*
surprise	*disgust*	*disgust*	*fear*	*neutral*	*fear*
surprise	*sadness*	*sadness*	*anger*	*joy*	sarcasm
surprise	*fear*	*fear*	*anger*	*surprise*	*anger*
surprise	*anger*	*anger*	*anger*	*disgust*	*anger*
surprise	*neutral*	*surprise*	*anger*	*sadness*	*anger*
sadness	*joy*	sarcasm	*anger*	*fear*	*anger*
sadness	*surprise*	*sadness*	*anger*	*anger*	*anger*
sadness	*disgust*	*disgust*	*anger*	*neutral*	*anger*
sadness	*sadness*	*sadness*	*neutral*	*joy*	*joy*
sadness	*fear*	*sadness*	*neutral*	*surprise*	*surprise*
sadness	*anger*	*anger*	*neutral*	*disgust*	*disgust*
sadness	*neutral*	*sadness*	*neutral*	*sadness*	*sadness*
disgust	*joy*	sarcasm	*neutral*	*fear*	*fear*
disgust	*surprise*	*disgust*	*neutral*	*anger*	*anger*
disgust	*disgust*	*disgust*	*neutral*	*neutral*	*neutral*
disgust	*sadness*	*disgust*			

Table A.1 Text and Emoji emotions combination mapping

Bibliography

[1] E. “Nell” Watson, *Humanizing Machines*. Cham: Springer International Publishing, 2019, pp. 233–246. [Online]. Available: https://doi.org/10.1007/978-3-030-16920-6_11

[2] I. B. A. TURING, “Computing machinery and intelligence-am turing,” *Mind*, vol. 59, no. 236, p. 433, 1950.

[3] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu *et al.*, “Towards a human-like open-domain chatbot,” *arXiv preprint arXiv:2001.09977*, 2020.

[4] R. K. Bakshi, N. Kaur, R. Kaur, and G. Kaur, “Opinion mining and sentiment analysis,” in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2016, pp. 452–455.

[5] B. Liu, “Sentiment analysis and opinion mining,” *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[6] N. Alswaidan and M. E. B. Menai, “A survey of state-of-the-art approaches for emotion recognition in text,” *Knowledge and Information Systems*, pp. 1–51, 2020.

[7] S. M. Mohammad, “Sentiment analysis: Detecting valence, emotions, and other affectual states from text,” in *Emotion measurement*. Elsevier, 2016, pp. 201–237.

[8] A. Seyeditabari, N. Tabari, and W. Zadrozny, “Emotion detection in text: a review,” *arXiv preprint arXiv:1806.00674*, 2018.

[9] P. Shaver, J. Schwartz, D. Kirson, and C. O’connor, “Emotion knowledge: further exploration of a prototype approach.” *Journal of personality and social psychology*, vol. 52, no. 6, p. 1061, 1987.

[10] “Rasa git repository,” 2020. [Online]. Available: https://github.com/RasaHQ/rasa

[11] P. Ekman, E. R. Sorenson, and W. V. Friesen, “Pan-cultural elements in facial displays of emotion,” *Science*, vol. 164, no. 3875, pp. 86–88, 1969.

[12] R. S. Lazarus and B. N. Lazarus, *Passion and reason: Making sense of our emotions*. Oxford University Press, USA, 1994.

[13] A. S. Cowen and D. Keltner, “Self-report captures 27 distinct categories of emotion bridged by continuous gradients,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 38, pp. E7900–E7909, 2017.

[14] T. W. Smith, *The book of human emotions: An encyclopedia of feeling from anger to wanderlust*. Profile books, 2015.

[15] J. A. Russell, “Culture and the categorization of emotions.” *Psychological bulletin*, vol. 110, no. 3, p. 426, 1991.

[16] M. Eid and E. Diener, “Norms for experiencing emotions in different cultures: Inter-and intranational differences,” in *Culture and well-being*. Springer, 2009, pp. 169–202.

[17] M. Wilhelm, “Wundt, grundzüge der physiologischen psychologie,” 1974.

[18] J. A. Russell and A. Mehrabian, “Evidence for a three-factor theory of emotions,” *Journal of research in Personality*, vol. 11, no. 3, pp. 273–294, 1977.

[19] S. Park, J. Kim, J. Jeon, H. Park, and A. Oh, “Toward dimensional emotion detection from categorical emotion annotations,” *arXiv preprint arXiv:1911.02499*, 2019.

[20] A. Mehrabian, “Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament,” *Current Psychology*, vol. 14, no. 4, pp. 261–292, 1996.

[21] H. Kessler, A. Festini, H. C. Traue, S. Filipic, M. Weber, and H. Hoffmann, “Simplex–simulation of personal emotion experience,” *Affective Computing*, pp. 255–270, 2008.

[22] B. Vlasenko and A. Wendemuth, “Location of an emotionally neutral region in valence-arousal space: Two-class vs. three-class cross corpora emotion recognition evaluations,” in *2014 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2014, pp. 1–6.

[23] D. Ghazi, D. Inkpen, and S. Szpakowicz, “Detecting emotion stimuli in emotion-bearing sentences,” in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed. Cham: Springer International Publishing, 2015, pp. 152–165.

[24] S. Buechel and U. Hahn, “EmoBank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 578–585. [Online]. Available: https://www.aclweb.org/anthology/E17-2092

[25] S. Mohammad and F. Bravo-Marquez, “WASSA-2017 shared task on emotion intensity,” in *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 34–49. [Online]. Available: https://www.aclweb.org/anthology/W17-5205

[26] S. M. Mohammad and S. Kiritchenko, “Using hashtags to capture fine emotion categories from tweets,” *Computational Intelligence*, vol. 31, no. 2, pp. 301–326. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12024

[27] D. Preoţiuc-Pietro, H. A. Schwartz, G. Park, J. Eichstaedt, M. Kern, L. Ungar, and E. Shulman, “Modelling valence and arousal in Facebook posts,” in *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 9–15. [Online]. Available: https://www.aclweb.org/anthology/W16-0404

[28] C. Strapparava and R. Mihalcea, “SemEval-2007 task 14: Affective text,” in *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 70–74. [Online]. Available: https://www.aclweb.org/anthology/S07-1013

[29] S. Aman and S. Szpakowicz, “Identifying expressions of emotion in text,” in *Text, Speech and Dialogue*, V. Matoušek and P. Mautner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 196–205.

[30] C. O. Alm, “Affect in text and speech,” 2009.

[31] V. Liu, C. Banea, and R. Mihalcea, “Grounded emotions, year=2017, volume=, number=, pages=477-483„” in *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*.

[32] “Crowdflower,” https://www.crowdflower.com/data/sentiment-analysis-emotion-text/.

[33] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu, "DailyDialog: A manually labelled multi-turn dialogue dataset," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 986–995. [Online]. Available: https://www.aclweb.org/anthology/I17-1099

[34] K. R. Scherer and H. G. Wallbott, "Evidence for universality and cultural variation of differential emotion response patterning." *Journal of personality and social psychology*, vol. 66, no. 2, p. 310, 1994.

[35] C. Alm, D. Roth, and R. Sproat, "Emotions from text: Machine learning for text-based emotion prediction." 01 2005.

[36] H. Schuff, J. Barnes, J. Mohme, S. Padó, and R. Klinger, "Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus," in *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 13–23. [Online]. Available: https://www.aclweb.org/anthology/W17-5203

[37] S. Mohammad, "#emotional tweets," in **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. Montréal, Canada: Association for Computational Linguistics, 7-8 Jun. 2012, pp. 246–255. [Online]. Available: https://www.aclweb.org/anthology/S12-1033

[38] H. Rashkin, E. Smith, M. Li, and Y.-L. Boureau, "Towards empathetic open-domain conversation models: A new benchmark and dataset," 01 2019, pp. 5370–5381.

[39] A. Chatterjee, K. N. Narahari, M. Joshi, and P. Agrawal, "SemEval-2019 task 3: EmoContext contextual emotion detection in text," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 39–48. [Online]. Available: https://www.aclweb.org/anthology/S19-2005

[40] S. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko, "SemEval-2018 task 1: Affect in tweets," in *Proceedings of The 12th International Workshop on Semantic Evaluation*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018. [Online]. Available: https://www.aclweb.org/anthology/S18-1001

[41] L.-A.-M. Bostan and R. Klinger, "An analysis of annotated corpora for emotion classification in text," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018. [Online]. Available: https://www.aclweb.org/anthology/C18-1179

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[43] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[45] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[46] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[47] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[48] D. You, O. C. Hamsici, and A. M. Martinez, "Kernel optimization in discriminant analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 631–638, 2010.

[49] A. McCallum, K. Nigam *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, no. 1. Citeseer, 1998, pp. 41–48.

[50] J. Rennie, L. Shih, J. Teevan, and D. Karger, "Tackling the poor assumptions of naive bayes classifiers (pdf)," *ICML.[Accessed: 10-Feb-2017]*, 2003.

[51] D. Subramanian, "Emoticon and emoji in text mining," https://medium.com/towards-artificial-intelligence/emoticon-and-emoji-in-text-mining-7392c49f596a, December 2019.

[52] D. Rajhi, "Emotional recognition using facial expression by emoji in real life," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, no. 9, 2007.

[53] P. K. Novak, J. Smailović, B. Sluban, and I. Mozetič, "Sentiment of emojis," *PloS one*, vol. 10, no. 12, 2015.

[54] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[55] C. Ma, H. Prendinger, and M. Ishizuka, "Emotion estimation and reasoning based on affective textual interaction," in *International Conference on Affective Computing and Intelligent Interaction*. Springer, 2005, pp. 622–628.

[56] "Rasa nlu data file," 2020. [Online]. Available: https://rasa.com/docs/rasa/nlu/training-data-format/

[57] "Rasa slots documentation," 2020. [Online]. Available: https://rasa.com/docs/rasa/core/slots/#id1

[58] "Rasa technologies: Custom nlu components," 2020. [Online]. Available: https://rasa.com/docs/rasa/api/custom-nlu-components/

[59] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "Diet: Lightweight language understanding for dialogue systems," *arXiv preprint arXiv:2004.09936*, 2020.

[60] M. Henderson, I. Casanueva, N. Mrkšić, P.-H. Su, I. Vulić *et al.*, "Convert: Efficient and accurate conversational representations from transformers," *arXiv preprint arXiv:1911.03688*, 2019.

[61] "Rasa core documentation," 2020. [Online]. Available: https://rasa.com/docs/rasa/core/about/

[62] "Rasa stories documentation," 2020. [Online]. Available: https://rasa.com/docs/rasa/core/stories/