ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**ARTIFICIAL INTELLIGENCE**

**MASTER THESIS**
**in**
**NATURAL LANGUAGE PROCESSING**

# Emotion Recognition for Human-Centered Conversational Agents

**CANDIDATE:**
**Luca Bolognini**

**SUPERVISOR:**
**Prof. Paolo Torroni**

**CO-SUPERVISOR:**
**Dr. Eleonora Mancini**

Academic year 2021/22
Session 3rd

# Abstract

This thesis proposes a study on Emotion Recognition in Conversation to address the challenges of the task with a chatbot reference case study to enhance conversational agents' ability to understand and respond appropriately to human emotion. The study consists of two phases. The first one involves the use of several baselines and the implementation of EmoBERTa to explore aspects of the task, such as preprocessing, balancing technique and context modelling tested on ERC benchmark dataset. The results reveal that the punctuation provides key information to the task, balancing techniques can provide marginal improvements if appropriately selected and context can provide additional information and suggest that a non-static context construction could be beneficial.

In the second phase, the effectiveness of a Few-Shot learning method, SetFit, is explored in the context of ERC to face the scarce amount of real labelled data. An incompatibility with the given context definition of the architecture employed by the mentioned method called for an adaptation which proved to be ineffective. The performance of the SetFit method and fine-tuning are compared in a limited data regime. Finally, the study explores the capabilities of a trained model on a specific ERC dataset to adapt to limited data from a different domain using Transfer Learning and fine-tuning with inconclusive results. The findings and insight from this can lay the groundwork for future developments and studies in the growing field of emotional-aware conversational agents and the application of Few-Shot learning in this task.

# Contents

# List of Figures

# List of Tables

# Introduction

In recent years, conversational agents have become increasingly popular as a way for businesses and individuals to interact with their customers, clients, or audience[1]. However, many conversational agents still lack the ability to understand and respond to the emotional states of their users, which can lead to frustration or disengagement. To address this limitation, researchers have started to explore the integration use of emotion recognition techniques in conversational agents[2].

Emotion Recognition (ER) refers to the ability of machines to identify and interpret human emotions based on various cues, such as facial expressions, tone of voice, or choice of words[3]. By incorporating emotion recognition into conversational agents, they can become more responsive and empathetic to the needs and feelings of their users[4, 5].

This thesis proposes a study of Emotion Recognition in Conversation (ERC) using a reactive chatbot as a reference case study. The chatbot serves as an example of how emotion recognition can be applied in practice, but the study is not limited to chatbots and our findings can be extended to other conversational agents or systems.

The study conducted in this thesis consists of two phases. The first phase focuses on implementing and testing our version of EmoBERTa, a state-of-the-art model for emotion recognition, and various baseline models to understand which preprocessing, data balancing techniques, and context structure are most effective to address ERC tasks. Architectures and techniques are evaluated employing benchmark datasets for ERC, namely MELD[6] and DailyDialog[7]. Results obtained in this phase serve to determine the experimental setting at the basis of the second phase.

The second phase addresses the problem of lack of real data for ERC by comparing different transfer learning approaches, such as few-shot learning and fine-tuning. Few-shot learning is machine learning strategy that allows a model to learn new concepts or tasks employing a small number of examples; on the other hand, fine-tuning is a process that takes a model that has already been trained for one given task and then tunes or tweaks the model to make it perform a second similar task. The study evaluates the effectiveness of few-shot learning and fine-tuning for ERC using a dataset of emotional conversations with limited labelled data. Results obtained using different approaches are comapred to determine the best one. Finally, the capabilities of a trained model on a specific dataset to adapt to a limited data from a different domain are explored using Transfer Learning and fine-tuning. Overall, this study contributes to the growing field of emotion-aware conversational agents and demonstrates the potential of few-shot learning and fine-tuning for emotion recognition in conversation. The findings and insights from this work can inform the development of future systems that prioritize user emotions and improve the quality of human-machine interactions.

The thesis follows this outline:

- *Chapter 1*: provides an overview of ERC, describing the task and its challenges. Popular datasets are then briefly described along with previous approaches. The Few-Shot Learning objectives and methods are introduced.
- *Chapter 2*: presents the datasets used during the experimental phase along with descriptions of their content and limitations.
- *Chapter 3*: contains the description of the architectures used as well as details and results of the experiments carried out.
- *Chapter 4*: offers a description of the limitations encountered and proposes possible future developments.
- *Chapter 5*: summarizes the work, its results and concludes.

# Chapter 1

# Background

In this chapter the tasks' objectives, challenges and datasets used are described and explored. The existing architectures and and generally useful background are introduced.

## 1.1 Emotion Recognition in Conversation

Emotion Recognition in Conversation seeks to develop algorithms and technologies to detect and interpret humans' emotional states. This aspect can provide information to implement more accurate and engaging human-machine interactions and it's crucial to create empathetic machines.

**Task Definition** This task's objective is to recognize the underlying emotion in dialogue utterances. A dialogue is a sequence of $N$ utterances, $d = [u_1, i_2, .., u_N]$. Each utterance is composed of $M_i$ words, $u_i = [w_1, w_2, .., w_{M_i}]$, spoken by a participant to the conversation, $S_i$, and will have a certain emotion connotation, $e_i$.

**Challenges** This task presents several challenges [8, 9]:

- *Emotion Taxonomy*: the first challenge to address is to define which are the emotions in the task and how they are described. To this end, several emotion models are proposed in the literature. We can divide them into two major categories: *categorical* and *dimensional*. The *categorical* models classify a definite number of discrete emotions. Ekman's emotion model[10] constitutes a standard for emotion description. It defines six fundamental emotions: *anger, sadness, joy, surprise, disgust* and *fear*. Plutchik's model[11], on the other hand, defines eight fundamental emotions organized in opposite pairs: *anger-fear, sadness-joy, anticipation-surprise, trust-disgust*. Furthermore, it defines emotion subtypes introducing different nuances of intensity. Figure 1.1 reports a representation of this model. Dimensional models adopt a continuous description of the emotional state. Usually, they use two of the three Valence-Arousal-Dominance(VAD) dimensions[12], namely valance and arousal. Valence describes the emotion's degree of positivity and Arousal its intensity. The dimensional representation allows a more comprehensive and comparable structure with respect to the categorical one. Different paradigms and taxonomies offer advantages and drawbacks. For example, simpler taxonomies, such as Ekman's, provide limited descriptive power with respect to Plutchik's. On the other hand, annotating data following the latter would be more complex because the discrimination between related emotions would be subtle. A more fine-grained taxonomy may induce in a lower agreement between the annotators due to its more complex nuances.

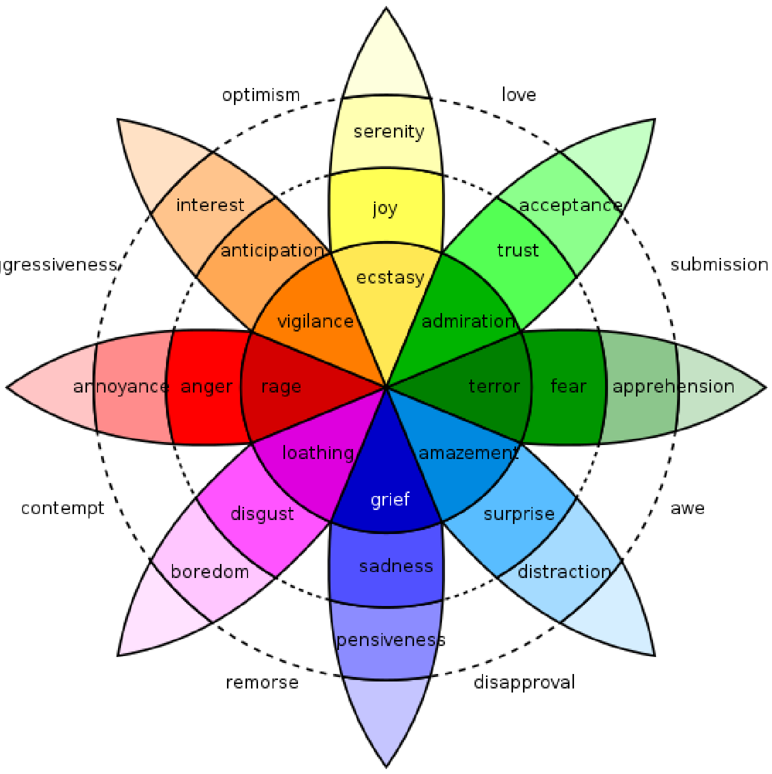

Figure 1.1: Plutchik's wheel of emotions.

- *Annotation*: the process of annotation presents its own challenges in emotion recognition, as it is a subjective task and therefore reflects individual interpretations of emotions. This subjectivity can result in differences in annotations among annotators, influenced not only by discrepancies in perception but also by personal and cultural backgrounds. Even when using the same taxonomy, the annotation process and resulting labels may differ. This is especially true for the annotation process of different datasets.

- *Context Modelling*: in ERC the context can provide key information to classify ambiguous utterances, especially short ones. Figure 1.2 reports an example of how two identical utterances can convey two different emotions according to the context. Contextual information can be extracted from the past and future utterances in the dialogue, depending on the field of application.

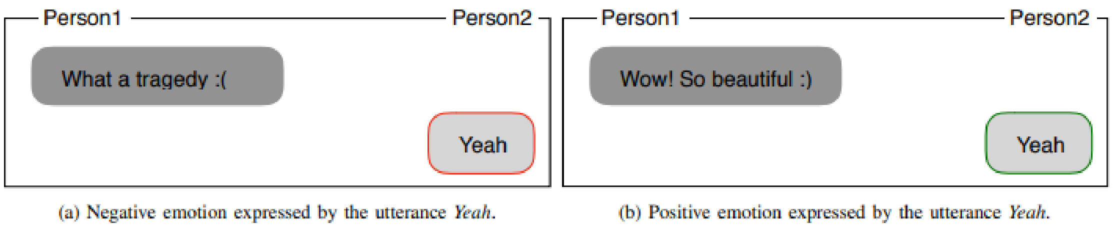
Person1
Person2
What a tragedy :(
Yeah
Person1
Person2
Wow! So beautiful :)
Yeah
(a) Negative emotion expressed by the utterance *Yeah*.
(b) Positive emotion expressed by the utterance *Yeah*.

Figure 1.2: Example of how the same utterance can express two different emotions according to the context.

- *Speaker Modelling*: the participants in the conversation have different personalities and ways of expressing themselves, distinct interpersonal relationships and dynamics, and each one has its mood. Speaker modelling exploits the speaker's utterance to profile them, gaining valuable information. This aspect becomes particularly relevant in multi-party conversations.

- *Dynamics Modelling*: emotion dynamics concern the evolution of the conversation participants' emotions throughout the dialogue. Two separate aspects are relevant in a conversation[13]: *self* and *inter dependencies*. *Self-dependency* describes the emotional inertia of one participant and how it influences itself. *Inter dependency* expresses how the participants affect each others' emotional state. Generally, emotional inertia tends to preserve the participant's current, but external stimuli may cause it to change, i.e. interacting with other speakers. These aspects are generally hard to model due to speakers' personality differences, conversational topics, and the nature of the participants' relationships.

| Emotion | EmotionLines | MELD | IEMOCAP | DailyDiaog |
|---|---|---|---|---|
| Neutral | 6530 | 6436 | 1708 | 85572 |
| Anger | 772 | 1607 | 1103 | 1022 |
| Disgust | 338 | 361 | - | 353 |
| Fear | 255 | 358 | - | 74 |
| Joy | 1710 | 2308 | 648 | 12885 |
| Sadness | 498 | 1002 | 1084 | 1150 |
| Surprise | 1658 | 1636 | - | 1823 |
| Frustrated | - | - | 1849 | - |
| Excited | - | - | 1041 | - |
| Total | 11761 | 13708 | 7433 | 102879 |

Table 1.1: Label distribution in different ERC datasets

**Datasets** In this paragraph, we briefly describe the datasets used as a benchmark in the task, reporting their source as well as their taxonomy. Table 1.1 reports their label distribution. It can be observed that the labels are generally unbalanced and this represents another big challenge of ERC.

- *IEMOCAP*[14]: a multi-modal dataset composed of *acted* dialogues. Its modalities comprehend audio, text and video. The emotion labels used to annotate the utterances are *happy, sad, neutral, angry, excited, frustrated, fear* and *disgust*.

- *EmotionLines*[15]: a textual dataset with dialogues gathered from the TV show *Friends* and Facebook private chats. Therefore the utterances are both *acted* and *spontaneous*. The emotion labels correspond to the six Ekman's basic emotions and the *neutral* class.

- *MELD*[6]: a multi-modal dataset composed of *acted* dialogues extracted from the TV show *Friends*. It is an enhancement of EmotionLines. The modalities comprise text, audio and video. The emotion labels correspond to the six Ekman's basic emotions and the *neutral* class.

- *DailyDialog*[7]: a large text dataset built from websites used to practice English dialogue in daily life. Therefore, the dialogues are *artificial*. Each utterance is labelled with one of the six Ekman's basic emotions or *other*.

### 1.1.1 Related Works

In this section, we provide an overview of the previous research done in this field. A wide variety of approaches and modalities have been explored, even if the textual modality is a constant throughout the literature. Many approaches involve the use of Convolutional Neural Networks (CNN) as feature extractors as initially proposed by [16] and LSTMs as context encoders. Several transformer-based techniques and graph-based networks are implemented.

- **DialogueRNN**[17]: introduces tracking of parties in the conversations. This architecture aims to model the main parts of the conversation: the speaker, the context of the previous utterance and the emotion evolution. To achieve this, they use three different GRUs:
  - The *Party GRU* encodes the speakers' states updated when the party speaks. These states allow tracking of their emotional dynamics.
  - The *Global GRU* encodes the states of the *preceding utterances* and the *party states*. Together they constitute the current utterance's *context representation*.
  - The *Emotion Representation* inferred by the model obtained using the current speaker's state and the preceding ones as context. The Emotion Representation of the previous utterance is used for the classification step.
- **DialogueGCN**[18]: the goal of this model is to capture self-emotional dependency (emotional inertia) and inter-speaker dependency.
  *Sequential Context Encoder*: a bidirectional GRU in which are fed the utterance representation $u_i$(obtained with the text-CNN presented by Kim) and the past and future states $g_{i\pm1}$, obtaining $g_i$: the context-aware utterance representation. This encoder is speaker agnostic.
  *Speaker-Level Context Encoder*: builds a graph where each node is the context-aware representation $g_i$ and the directed edges represent either speaker or temporal dependency. To reduce computational costs, the authors define a number of past (*p*) and future(*f*) utterances that can share an edge (e.g., $[v_{i-p}, .., v_i, .., v_{i+f}]$). They use transformations on the graph information and concatenate the result with the $g_i$ representation, apply an attention layer and feed the result to FC for the classification.
- **EmoCaps**[19]: the dialogues are into a series of single utterances that are then fed into BERT and retrieve the embedding vectors. The authors highlight the use of three factors

mainly: the utterances' *emotional tendencies* (which is something close to an offset that skews the emotion weight from the neutral class), different emotional information from the three modalities, *contextual information*. They build an emotion vector which is then merged with a sentence vector. They use *residual learning* in the construction of the *EmoFormer*, which extracts the emotion vector. According to one author, this project is mainly focused on the textual modality.

- **EmoBERTa**[20]: RoBERTa is used as a feature extractor. The peculiarity lies in the context definition. It consists of creating an input sequence containing both the current and the context utterances. The architecture relies on RoBERTa's capacity of handling three different segments. Furthermore, the authors prepended the speaker's name to each utterance.

- **M2FNet**[21]: a multi-modal hierarchical framework designed on two levels of utterance and dialogue level feature extraction. Firstly, each modality's features are extracted from the utterances. Then, at the dialogue level, the model learns to label each utterance using the contextual information from the entire dialogue. The text feature extractor is *RoBERTa* and uses the context definition given by [20]. Each input utterance is concatenated with the preceding and following utterances. The embeddings obtained from each modality are passed to their corresponding network with a variable stack of transformers encoders and merged for the classification.

- **EmotionFlow**[22]: the context is defined through the *k* previous utterance and the corresponding speaker. This method highlights the emotion interconnection between two consecutive utterances and how likely is an emotion to shift to another (e.g. they report that a shift from *sadness* to *fear* is unlikely). A prompt question (*"How does Speaker$_x$ feel?"* is concatenated to the last *k* speaker-utterance pairs. The result is then fed into RoBERTa. A Conditional Random Field layer is used to maximize the probability of the ground truth emotion sequence over all possible sequences.

Table 1.2 reports the performances of the approaches listed above on MELD and IEMOCAP datasets.
Considering our task, we decided to limit the scope of our work to the textual modality. As [21] reports in their ablation studies, the textual modality provides the most relevant information to the task. The context definition defined by [20] provides a starting point for context studies and

| Network | Datasets | Macro-F1 | W-F1 |
|---|---|---|---|
| DRNN | I | 61.21/- | 62.75 |
| DGCN | I/M | 64.18/- | 64.18/58.10 |
| EmoBERTa | I/M | -/- | 68.57/65.61 |
| M2FNet | I/M | -/- | 66.38/66.23 |
| Emocaps | I/M | 72.60/44.08 | 71.77/64.00 |
| EmotionFlow | M | - | 65.05 |

Table 1.2: Performances of the approaches presented on MELD (M) and IEMOCAP (I).

will be explored in the experimental phase. Most of these works do not produce information about preprocessing steps.

## 1.2 Few-Shot Learning

Few-shot learning (FSL) is a Machine Learning framework[23]. It aims to train a model using a limited number of supervised data samples while maintaining competitive performances. There can be multiple reasons to develop this research field:

- *Learning rare Cases*: the data about certain supervised information might be hard or impossible to gather due to privacy and ethical issues, for example. FSL can learn models for rare cases.
- *Reducing costs and effort*: gathering and annotating data is resource and time-consuming. Reducing the number of samples required to train an efficient model relieves part of the burden.
- *Deal with unbalanced data*: the need for a limited number of samples to train the model implies that unbalances in the class distributions of a dataset can be bypassed using a balanced subset.

The reasons listed above are all relevant to the ERC task. The amount of real data gathered in the literature so far is limited, and acquiring it may be hindered by privacy issues, especially in a chatbot framework. Furthermore, the Emotion Recognition task is amongst the most subjective tasks making complex annotating a large corpus. Reducing the quantity of data necessary to train models is desirable. Finally, as mentioned in section 1.1, the datasets available present generally unbalanced label distribution. FSL could be another tool to use as a balancing technique.

Formally, a few-shot classification problem considers a *N-way K-shots* scenario, in which a training set D$_{train}$ contains $S = K \times N$ samples where we have *N* classes each with *K* examples. In this scenario D$_{train}$ is small and training a model able to approximate a good generalization becomes difficult. Two different paradigms can be defined: *Meta-learning-based* FSL and *Non Meta-learning-based* FSL[24]. The first paradigm can be classified into three approaches:

- *Metric-based*
- *Optimization-based*
- *Model-based*

In the following paragraph we report information on metric-based approaches in which the model learns a distance function and leverages the meta-learning architecture to learn an embedding function. Among the architectures in the literature, we report the following:

- *Matching Networks[25]*: these networks define a probability distribution over the output labels using an attention kernel. This kernel computes a cosine similarity between the embeddings of support and query examples and the result is normalized with a softmax. The classification output is defined as the sum of support samples weighted by the attention kernel.
- *Siamese Networks[26]*: this architecture employs two networks with shared weights and through the use of a distance metric, such as Euclidian Distance, it learns to recognize if two examples belong to the same class.
- *Prototypical Networks[27]*: for each class, the support examples are embedded and the averaged to create the class's prototype. A similarity score is then used to compute the distances between the query's embeddings and the prototypes. The output probability is calculated by applying a softmax to the negative distances.

As far as the second paradigm's approaches are concerned, among them we can find Transfer Learning[28], which exploits the knowledge learned from a certain task A to enhance the modelling of a task B.

### 1.2.1 Related Works

This section describes some approaches and method for FSL related to ERC and other tasks. In [29], the authors propose an architecture for FLS in ERC using a Text-CNN as the utterances feature extractor followed by a BiLSTM to capture the dialogue's contextual information. This part of the model is the context encoder. Its output is fed to 2 Fully Connected layers which produce the representation used to compute the prototypes. Finally, to capture dependencies between the labels they introduce a CRF layer. They use episodic learning[30] to train the model which is tested on DailyDialog and a dataset proposed by them composed of French customer service chats. The model's evaluation on DailyDialog's test set is carried out using 1000 random samples, making it not comparable with our experiments. The F1-weighted they produce is 0.3522.

The authors of [31] propose an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers (ST). This method finetunes a pretrained ST using a small number of text pair, in a contrastive Siamese manner. The resulting model is able to produce meaningful embeddings used to train a classification head. This strategy has been applied to several tasks, Emotion Recognition included, achieving good results. We choose this simple framework as a starting point for our FSL experiments to test its effectiveness in ERC. For further details on the training procedure and the architecture employed, refer to section 3.2.1.

# Chapter 2

# Datasets

This section describes the characteristics and limitations of the datasets used in the experiments. We illustrate the data's source and the modality present in the dataset, the features structure and the taxonomy used. Finally, statistics about the utterances, dialogues and label distribution are reported.

## 2.1 Multimodal EmotionLines Dataset - MELD

Multimodal EmotionLines Dataset[6] (MELD) is a dataset composed of dialogues extracted from the *Friends* TV show. It contains annotations to address both the Emotion Recognition and Sentiment Analysis tasks. The dataset encompasses three different modalities: textual, audio, and video. It is an expansion and enhancement of the EmotionLines dataset[15], which comprises only the textual modality. The annotations have been improved thanks to the additional information provided by audio and video. Moreover, the dialogues have been filtered, removing the utterances that were not compliant with constraints imposed on their starting and ending times.

As far as the ERC task is concerned, each utterance is annotated according to Ekman's emotion model mentioned in section 1.1. Therefore, the emotion labels are *Anger, Disgust, Fear, Joy, Surprise* and *Sadness*. The utterances considered not to contain any emotional value are labelled as *Neutral*.

On the other hand, the sentiment labels are given using an aggregation method:

- The negative label is associated with anger, disgust, fear and sadness.
- The positive label is associated with joy.

- The neutral label to the neutral emotion labeled utterances.

The sentiment associated with surprise is evaluated for each utterance since it can be either positive or negative. Table 2.1 reports the number of samples for each emotion across the *train, validation* and *test* splits given by the authors of the dataset. Amidst these divisions, the mean length of each utterance - that is, the number of words within each sentence - is practically identical. Similarly, the amount of emotions in each conversation is also almost equivalent. Table 2.3 reports handy statistics about the utterances and dialogues.

| Emotion | Train | Val | Test |
|---|---|---|---|
| neutral | 4710 | 470 | 1256 |
| joy | 1743 | 163 | 402 |
| surprise | 1205 | 153 | 345 |
| anger | 1109 | 150 | 281 |
| sadness | 683 | 111 | 208 |
| disgust | 271 | 40 | 68 |
| fear | 268 | 22 | 50 |
| Tot. | 9989 | 1109 | 2610 |

Table 2.1: Label distribution across MELD's splits.

The dataset's features are:

- *Sr No*: serial number associated to the utterance.
- *Utterance*: individual utterance reported as a string.
- *Speaker*: the utterance speaker's name.
- *Emotion*: the emotion expressed by the speaker. This feature represents the label used in our task.
- *Sentiment*: the emotion's polarity (*positive, neutral, negative*)
- *Dialogue ID*: utterance's dialogue.
- *Utterance ID*: the utterance's index in the dialogue.
- *Season*: utterance's season.
- *Episode*: utterance's episode.
- *StartTime*: utterance's begin time in the episode.

- *EndTime*: utterance's end time in the episode.

Table 2.2 reports a snippet of the dataset.

| Utterance | Speaker | Emotion | Sentiment |
|---|---|---|---|
| Can I tell you a little secret? | Phoebe | neutral | neutral |
| Yeah! | Rachel | joy | positive |
| I want to keep one. | Phoebe | neutral | neutral |
| "Ohh, I'm gonna be on the news!" | Rachel | joy | positive |

Table 2.2: Example of a dialogue extracted from MELD's test set.

This dataset was selected because is a popular benchmark in ERC. The multi-modal nature offers the opportunity to continue the work integrating other modalities in future endeavours and, while its multi-party conversation presents a challenge, it provides a more general setup.

### 2.1.1 Bias based on the speaker's personality

There are 304 different speakers that make an appearance in the dataset. The main six characters hold most of the utterances (about 83%) as shown in Figure 2.1. The unbalance in the speakers' distribution might introduce a bias in the models through the characters' interaction, dynamics and personalities, even if the speakers' names are omitted. We cannot avoid this bias because the characters' interactions are deeply intertwined. Therefore, removing characters from one split to prevent information leakages into other splits would result in deleting most of the available data.

### 2.1.2 Unbalanced Data

Exploratory Data Analysis revealed that the dataset is unbalanced. This is particularly evident by analyzing the label count. The neutral class is the most populated one while the emotional classes are less frequent, especially the negative ones. A possible reason for this disparity could lie in the data origins because the dialogues are extracted from a comedy show, it can be expected the negative emotions to be the minority. However, the unbalance issue is related to the task: it is clearly stated in the literature that other benchmarks [7, 14, 15, 32, 33] for ERC tasks suffer this phenomenon as well. A hypothesis could be that the utterances are extracted from dialogues, which generally contain a non-uniform emotion distribution. Therefore, this unbalance can be expected even in data collected in real scenarios. In [29], authors present a French

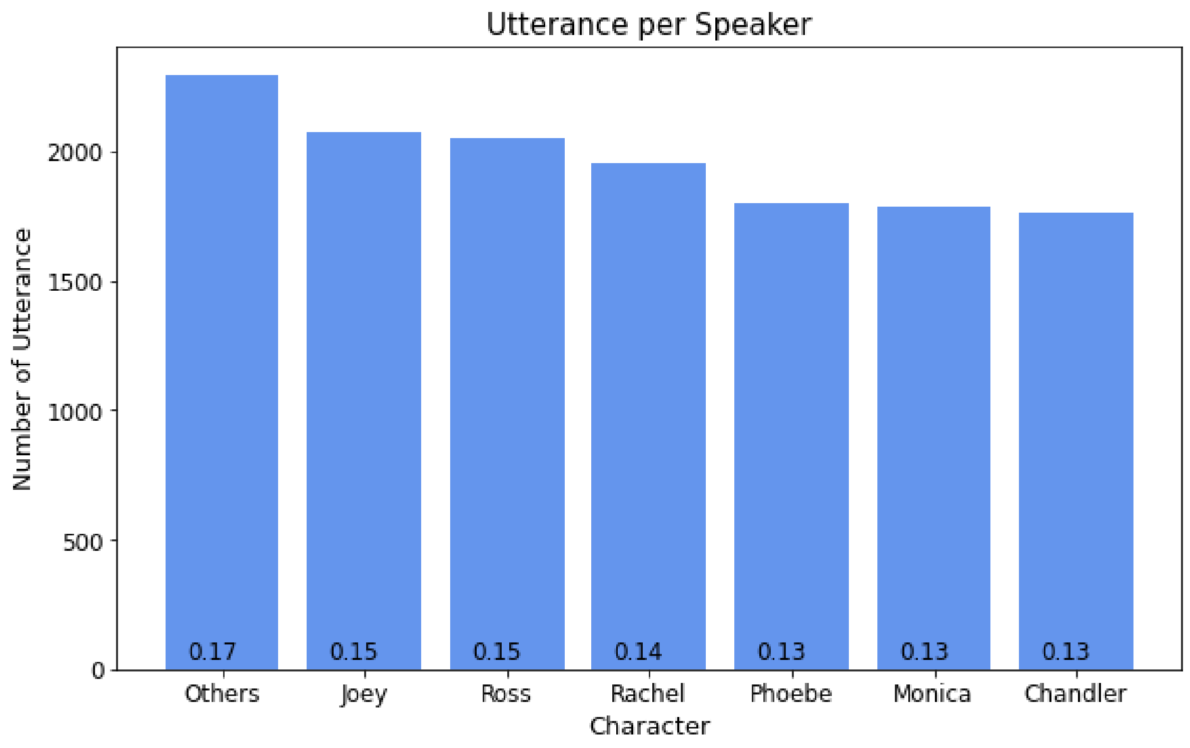

Figure 2.1: MELD number of utterances per speaker.

| MELD Statistics | Train | Validation | Test |
|---|---|---|---|
| Avg. utterance length | 8.0 | 7.9 | 8.2 |
| Max utterance length | 69 | 37 | 45 |
| Number of dialogues | 1039 | 114 | 280 |
| Number of utterances | 9989 | 1109 | 2610 |
| Number of speakers | 260 | 47 | 100 |
| Avg. number of utterances/dialogue | 9.6 | 9.7 | 9.3 |
| Avg. number of emotions/dialogue | 3.3 | 3.3 | 3.2 |
| Number of emotion shift | 4003 | 427 | 1003 |

Table 2.3: MELD Statistics

dataset of collected private chats and the same phenomenon occurs. During the experimental phase, we tested potential approaches to this issue (refer to section 3.1.3.2).

## 2.2 DailyDialog

DailyDialog[7] is a multi-turn dialogue dataset. The raw data are extracted from websites used for English learners to practice in daily dialogues, hence the name. The conversations are artificial and encompass various daily life topics and contexts such as an interaction between a customer and a shopkeeper or two students. DailyDialog comprises only the textual modality and. Ekman's emotion model is used as the taxonomy for the annotation with the addition of a *other* class, which for our purposes we associate with MELD's *neutral* class. The conversation topics are clustered into 10 categories. The largest three are *Relationships*, *Ordinary life* and *Work*. Furthermore, a label indicating the intention behind an utterance is annotated. The *act* classes are:

- *Inform*: statements and question that provide information.
- *Question*: utterances that seeks information.
- *Directives*: requests, instructions and suggestions.
- *Commissive*: reactions to requests and offers.

| Utterances | Dialogue_ID | Utterance_ID | Emotion | Act |
|---|---|---|---|---|
| The taxi drivers are on strike again. | 1 | 0 | neutral | inform |
| What for? | 1 | 1 | neutral | question |
| They want the government to reduce [..] | 1 | 2 | neutral | inform |
| It is really a hot potato. | 1 | 3 | neutral | inform |

Table 2.4: Dialogue example extracted from DailyDialog's test set.

The data are stored in text files that need to be parsed to obtain a structured dataset. Hence, we exploit this operation to retrace MELD's structure adding Dialogue IDs and Utterances IDs as defined in section 2.1.
The dataset's features are:

- *Utterance*: individual utterance reported as a string.
- *Act*: the intention behind the utterance.
- *Emotion*: the emotion expressed in the utterance.
- *Dialogue ID*: dialogue identifier.
- *Utterance ID*: utterance's index in the dialogue.

Table 2.4 is an example of a dialogue extracted from the dataset. Table 2.5 reports the number of samples for each emotion across the *train, validation* and *test* splits provided by the authors of the dataset. It can be observed that the label distribution is heavily unbalanced, as it was also mentioned in section 2.1.2, and the neutral class is the most populated class by a large margin. Table 2.6 reports an overview of the sets' statistics.

| Emotion | Train | Val | Test |
|---|---|---|---|
| neutral | 72143 | 7108 | 6321 |
| joy | 11182 | 684 | 1019 |
| surprise | 1600 | 107 | 116 |
| sadness | 969 | 79 | 102 |
| anger | 827 | 77 | 118 |
| disgust | 303 | 3 | 47 |
| fear | 146 | 11 | 17 |
| Tot. | 87170 | 8069 | 7740 |

Table 2.5: Label distribution across DailyDialog's splits.

| DailyDiaog | Train | Validation | Test |
|---|---|---|---|
| Avg. utterance length | 13.6 | 13.5 | 13.8 |
| Max utterance length | 278 | 166 | 204 |
| Number of dialogue | 11118 | 1000 | 1000 |
| Number of utterances | 87170 | 8069 | 7740 |
| Avg. number of utterances/dialogue | 7.84 | 8.07 | 7.74 |
| Avg. number of emotions/dialogue | 1.58 | 1.56 | 1.61 |

Table 2.6: DailyDialog Statistics

This dataset was selected as MELD's counterpart in later experiments. The taxonomy used in the annotation process is the same one used in MELD, but the different text structures and contents can provide the setup for the experiments on few-shot learning in another domain.

# Chapter 3

# Experiments

## 3.1 Experiments on ERC

This section includes descriptions of the models' architectures, as well as the details of the experiments conducted, and the results obtained. In addition, analysis and hypotheses are formulated on the behaviours witnessed.

### 3.1.1 Evaluation metrics

The metrics that we used to evaluate the performances are the macro and weighted *F1-score*. The former is the unweighted mean of the F1 scores. It reflects how the model truly fares in the task, giving the same importance to every class and limiting the skew due to the unbalances present in the data. The latter considers the F1-score of each class weighted with its support. As a consequence, the more populated classes, such as the *neutral* one in our case, will have greater importance when computing the score.

### 3.1.2 Architectures

#### 3.1.2.1 Baselines

Few baselines have been tested on MELD:

- Most Frequent Classifier.
- Random Uniform Classifier.
- NRCLexicon[34].

- Text-CNN[16].

**Random Baselines** The first two baselines make predictions relying only on the number and the frequency of labels ignoring the input features: the uniform baseline generates the prediction considering each label equally probable. Conversely, the most frequent baseline predicts that every sample has the same label as the dataset's most populated class.

**Lexicon** The lexicon baseline uses the knowledge gathered in the NRCLexicon. This tool contains associations between words and a set of eight emotions, namely *anger, fear, anticipation, trust, surprise, sadness, joy, and disgust*. For example, the word *abundance* is associated with joy and trust emotions. When an utterance is analyzed using the lexicon, it builds a list of emotions according to the words and their associations. The most frequent ones, tied to an equal number of occurrences, are stored in a *top emotion* list. The utterance's label is compared with the *emotion* list to generate a prediction from this tool. The prediction is correct if the label is in the list or the list is empty when the label is *neutral*. Otherwise, the prediction is considered wrong. This method generates a *recall* score.

**Text-CNN** The Text-CNN has been used as a feature extractor in several ERC pieces of research [17, 18, 29, 35, 36]. This model's architecture, in our implementation, comprises an embedding layer in which we upload the embedding matrix. The embedded input is reshaped and given to 3 convolutional layers with 3,4 and 5 as filter sizes respectively. Each layer has 50 filters. The feature maps produced are global max pooled and concatenated to be fed to a dense layer. The output of this process is a vector representing the textual features which are then given as input to a *softmax* activated layer to produce the classification prediction. Table 3.1 summarises this architecture.
The following paragraph describes the operations executed to prepare an input the neural network can process. Firstly, the utterances are preprocessed, and then we add a blank space between each word and punctuation mark, if any, to avoid spurious encoding. To avoid information leakages between the splits, we build the embedding matrix following these steps for each split:

- We create a vocabulary containing all the unique words in the split. If it already exists, we expand it with unseen words.

- The co-occurrence matrix is built on the split's utterance. This matrix contains the number of times two words occur within a fixed window in the utterances. This window defines a neighbourhood.
- The embedding matrix is built and expanded by associating each new word's index to the corresponding pre-trained embedding vector from an embedding model. When out-of-vocabulary (OOV) words are met, namely the words not present in the embedding model, we compute the mean of the neighbouring words' embeddings. If this is not possible, we initialize the vectors randomly.

The utterances are then encoded, substituting each word and punctuation mark to the corresponding index in the vocabulary. Simultaneously, we pad and truncate the utterances to the desired maximum length. We use GloVe[37] as embedding model with 300 as the embedding dimension. The embeddings are kept static throughout the training. The OOV words are around 3.94% of the unique words found. Using the mean of the 4 neighbouring words, we compute their embeddings. As mentioned in the previous section, to discern which preprocessing is the most appropriate and gain insight into which textual elements are more relevant to the ER task, we use this model to test different preprocessing styles. We report the results in section 3.1.4.1.

#### 3.1.2.2 RoBERTa

One of the architectures used during this experimental phase is RoBERTa[38]. This pre-trained language model is based on BERT's architecture but uses a different training setting, it is composed of multiple layers of transformer encoders as it was defined by [39]. The number of layers and their hidden sizes change according to the model version, either *base* or *large*. BERT takes as input a concatenation of two segments (sequences of tokens), $x_1, .., x_N$ and $y_1, ..., y_M$ which usually consist of more than one natural sentence. The two segments are presented as a single input sequence to BERT with special tokens delimiting them:

$$[CLS], x_1, ..., x_N, [SEP], y_1, ..., y_M, [EOS] \tag{3.1}$$

Two different versions of this architecture are available RoBERTa$_{base}$ and RoBERTa$_{large}$. They are composed of multiple layers of transformer encoders as it was defined by [39]. The architectural difference lies in the structural parameters such as the hidden layers' number and their size, namely 768 for 12 layers in the *base* model and 1024 for 24 layers in the *large* version,

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input (InputLayer) | [(None, 10)] | 0 |
| embedding (Embedding) | (None, 10, 300) | 2298000 |
| reshape (Permute) | (None, 300, 10) | 0 |
| Conv_filter_3 (Conv1D) | (None, 298, 50) | 1550 |
| Conv_filter_4 (Conv1D) | (None, 297, 50) | 20508 |
| Conv_filter_5 (Conv1D) | (None, 296, 50) | 2550 |
| global_max_pooling1d (GlobalMa xPooling1D) | (None, 50) | 0 |
| concatenate (Concatenate) | (None, 150) | 0 |
| dense (Dense) | (None, 100) | 15100 |
| dropout (Dropout) | (None, 100) | 0 |
| dense_1 (Dense) | (None, 7) | 707 |

Total params: 2,319,957
Trainable params: 21,957
Non-trainable params: 2,298,000

Table 3.1: Model architecture used for Text-CNN Model

and ultimately in the number of total parameters in the model, 125M and 355M. This model is used as a feature extractor from the textual input. The pooled output corresponding to the [CLS] token is then fed to a Fully Connected layer and the classification is performed through a FC with softmax activation function. In all experiments described in section 3.1.3.2, we used mainly the larger model[1].

[1] https://huggingface.co/roberta-large

### 3.1.3 Analyzed Aspects

This section describes the dimension explored divided according to the used architecture.

#### 3.1.3.1 Aspects analyzed with Baselines

**Preprocessing** Data preprocessing is an essential step in every Machine Learning application aimed at transforming raw into clean data. In NLP data processing techniques are mainly used to remove unwanted text components that can be detrimental to the task at hand or to standardize the characters present in the corpus. To enhance model deployment and ensure robustness in the face of diverse real-world data, a crucial step is the effective cleansing of input data, which allows for a reasonable expectation that the textual components of the data seen during training will be reflected. In the present experimentation, various preprocessing approaches were explored to determine their efficacy and potential drawbacks for the task at hand.

#### 3.1.3.2 Aspects analyzed with RoBERTa

**1. Context**

As mentioned in section 1.1, it is possible to leverage contextual information. The authors of [20] have designed an easy way to input context information along with the utterances that have to be classified. Such a method consists of prepending and appending the past and future utterances to the currently considered utterance. This method exploits the computational power and the pretraining of the RoBERTa architecture[38] which is capable to handle sentences divided into three different segments delimited by the separator token *'</s></s>'*. This special sequence of characters is associated with a specific meaning during the pretraining phase of the language model: </s></s> is used as a separator between segments, [SEP], </s> is used as end of sequence, [EOS], and <s> is used as classification token, [CLS].
The input built in this way will have the following structure:

”<s>past_utterances</s></s> current_utterance </s></s> future_utterances</s>” (3.2)

Considering the context of this thesis's application, i.e. a chatbot, it is not possible nor it would appropriate to consider future utterances because they are not known beforehand, especially in a real time scenario. Therefore, to training the model with this input structure would not be meaningful because it is a scenario which would never be encountered once the model is deployed. As a consequence, the input comprises only a number of past utterances. To determine how variations of this number affect the performances, it is necessary to investigate its influence on the predictions' accuracy. It must be noted that the contextualized input should be built before applying any of the balancing techniques discussed in the section 3.1.3.2, otherwise, the dialogue structure present in the dataset could be altered, losing key information about its unfolding.

To build the contextualized input, the dataset structure is exploited: each dialogue is processed and identified through the ID and its utterances are considered to be sequential and to each one is given a new ID. This step was necessary due to the fact the given IDs are not always consecutive and the missing values would create a gap hindering the process. The utterances with ID in the range $[current\_utterance\_ID - number\_past\_utterances, current\_utterance\_ID]$ are concatenated into a context string which is then prepended along with the [SEP] to the current utterance. It must be noticed that there are spurious contexts: if we consider 3 past utterances as context, for example, the first 3 utterances will have a different number of past utterances prepended because there are not enough utterances spoken in the dialogue yet. The model is able to handle this difference due to its pretraining and because the context is defined as a sequence and there is no special character to separate its different utterances. The first utterance of each dialogue will always be context-less by definition and they will remain unmodified. It was tested if the addition of a blank space or an empty string followed by the separator improved the model performance but they were either unaffected or slightly worse.

Considering the dialogue reported in Table 2.2 the respective contextualized utterances with one past utterance are reported in Table 3.2. To achieve this goal each dialogue is processed and each utterance is prepended to the chosen number of past utterances. The result is stored as a new feature of the dataset and will be used as input for the model training.

| ID | Current utterance | Contextualized utterance |
|---|---|---|
| 0 | Can I tell you a little secret? | Can I tell you a little secret? |
| 1 | Yeah! | Can I tell you a little secret?</s></s> Yeah! |
| 2 | I want to keep one. | Yeah!</s></s>I want to keep one. |
| 3 | ”Ohh, I'm gonna be on the news!” | I want to keep one.</s></s>”Ohh, I'm gon... |

Table 3.2: Example of the current and contextualized utterance

| Emotion | Emotion Count | Weight |
|---|---|---|
| neutral | 4710 | 0.303 |
| joy | 1743 | 0.819 |
| surprise | 1205 | 1.184 |
| anger | 1109 | 1.287 |
| sadness | 683 | 2.089 |
| disgust | 271 | 5.266 |
| fear | 268 | 5.325 |

Table 3.3: Class weights computed on the MELD's training set

**2. Balancing Techniques**

As mentioned in section 2.1.2, the available datasets are quite unbalanced. Several techniques can be employed to minimize the effects of the different class distributions without discarding any data, due to the limited amount available.
The techniques that could be tested are:

- *Classe Weights*
- *Resample*
- *Adversarial Augmentation*

**Class Weights** One of the techniques tried is using *class weights* computed with the *sklearn* library. This method computes class weights that are then applied to the loss function during training in order to reduce the prediction skew of the model towards the most populated label in the dataset. This technique's objective is to magnify the effects of predictions for the less populated classes and reduce the weight of the predictions of the most populated ones. The weights computed with the *balanced* mode offered by *sklearn* will reflect the frequency of the class they are associated with. The less populated class will have a bigger weight on the loss function while, on the other hand, the most frequent classes will generally have smaller weights (possibly smaller than one). Table 3.3 reports the weights computed on MELD's training set. It's worth noting that this method improves the recall of the less populated classes but also reduces the precision of their prediction and the recall of the other classes is worse. The overall F1-scores (weighted and macro) are either unchanged or worse.

**Resample** The second technique applied is *resampling*. This strategy can be performed by *upsampling* the least populated classes by injecting copies of certain samples already present in the training split multiple times or by *downsampling* the most populated classes removing examples. This method has to be used with caution because it can reduce the generalization capabilities of the model, increasing overfitting chances due to the reduced diversity in the training examples. Given MELD class distribution, refer to section 2.1.2, it is desirable to increase the number of utterances with the *fear, disgust* and *sadness* labels. It would be also possible to *undersample* the *neutral* class, by removing the samples labeled as such, but considering the limited amount of data available it was undesirable. A test could be to combine the two techniques and compute the weights on the upsampled dataset: we would still benefit from the upsampling's effects and the weights computed would be more uniform, imposing a less severe penalty on the neutral class.

**Adversarial Augmentation** Adversarial Augmentation presents a technique used to tarnish textual data by introducing grammatical errors, removing words, and swapping word positions. It is possible to use the *TextAttack*[40] library to apply this strategy. This process could be used to introduce noise in the available clean data, bringing them closer to the real data in the case study we are analyzing. The messages exchanged with a chatbot might contain grammatical errors, such as typos and word misspellings, and missing or wrongly automatically corrected words[29]. This technique's efficacy cannot be evaluated on the available benchmark dataset without altering the test sets, therefore making the results incomparable with other models. Furthermore, to confirm the validity of this approach real data should be gathered and inspected.

### 3.1.4 Experimental Setup and Results

#### 3.1.4.1 Baseline Experiments

**Experimental Setup**

The first experiments were done with *most frequent* and *uniform* baselines and the Text-CNN. It must be noted that these baselines do not consider any contextual information from other utterances in the dialogue. The *most frequent* and *uniform* baselines do not require any particular setup because they only consider the label distribution of the training set, the first one, and the number of labels in the second one. The lexicon baseline uses the method we described in section 3.1.2.1 to generate the prediction. As mentioned in the 3.1.3.1, with the Text-CNN several different text preprocessing are tested to find the most appropriate one. The differences in the preprocessing steps mainly concerned punctuation. The utterances' text was

| Preprocess | F1-Weighted | F1-Macro | Preprocess | F1-Weighted | F1-Macro |
|---|---|---|---|---|---|
| 1 | 0.36338 | 0.15704 | 1 | 0.34967 | 0.13908 |
| 2 | **0.48703** | **0.25717** | 2 | **0.45221** | **0.22181** |
| 3 | 0.46323 | 0.23496 | 3 | 0.43739 | 0.21192 |
| 4 | 0.47230 | 0.23886 | 4 | 0.44661 | 0.21032 |

Table 3.4: Text-CNN Scores on MELD Test split with different preprocessing, best results highlighted in bold, max input length set to longest in the training set (left), max length set to 10 (right). The preprocess used correspond to the ones enumerated in 3.1.4.1.

always standardized and lowered and the contractions expanded throughout the experiments. The different preprocessing configurations analyzed are the following:

1. The text was stripped of all punctuation.
2. All punctuation removed except for question and exclamation marks.
3. All punctuation removed except for question and exclamation marks, dots and commas.
4. All punctuation kept.

In these experiments, the model was trained with both the maximum length of the input equal to 10, i.e. slightly above the training set average, and with the maximum length equal to the longest utterance in the training split.

**Results** Table 3.4 reports the preprocessing experiments' results. By inspecting them, we can observe that the punctuation characters provide salient information to the model, especially question and exclamation marks. Intuitively, these punctuation marks are indicators of a particular emotive inflexion of the utterance otherwise undetectable by the model. The differences between removing or keeping the other punctuation marks are marginal. Considering these results, we choose to keep all the punctuation and only standardize the characters in the utterance text because several different punctuation characters represent the same one (i.e. different symbols to represent dots, commas, dashes, apostrophes, etcetera) without removing any of them.
Table 3.5 reports the baselines' F1-scores on MELD's test set. We use the best Text-CNN score, i.e. the best score obtained in the preprocessing experiments.

| Model | F1-Weighted | F1-Macro | Macro recall |
|---|---|---|---|
| Most-Frequent | 0.31 | 0.09 | 0.14 |
| Uniform | 0.17 | 0.11 | 0.13 |
| Lexicon | - | - | 0.19 |
| Text-CNN | 0.487 | 0.257 | 0.25 |

Table 3.5: Baseline Scores on MELD Test split

#### 3.1.4.2 RoBERTa Experiments

This section describes the experiments carried out using the RoBERTa architecture and the context definition given in 3.1.3.2, namely the context exploration and balancing technique, and reports the results that emerged.

**1. Context Experiments**

**Experimental Setup** To find the optimal number of utterances to build the context and evaluate their effects on the performances, we train the model with several past sentences, namely between 0 and 4. The model used to carry out these experiments is the one described in 3.1.2.2. The maximum length of the input is fixed to 90 tokens. We made this choice to keep the training time moderate without truncating a significant amount of samples (only 5% of the utterance are truncated with 4 past utterances). We use the RobertaTokenizer[2] to encode and tokenize the utterances. The padding and truncation are set on the left side of the input string to maintain the current utterance unmodified.

**Results** Table 3.6 displays these experiments' results. We can observe that introducing even a single past utterance enhances the performance; however, when utilizing more than one utterance as context, the improvement is less significant. These outcomes might suggest that the preceding utterance is more closely linked to an emotional shift in the speaker and is generally the most relevant to the current utterance at the content level. The table also shows that using two past utterances leads to worse performance compared to the case with one past utterance. These findings might imply that emotional shifts occur more frequently directly in response to other participants in the conversation. These results might suggest that the additional context utterance increases the noise in the input rather than bringing new information. Moreover, the dataset's non-dyadic nature might be involved because other speakers may intervene in the conversation with unrelated sentences. Another possible interpretation of these results is that

[2]https://huggingface.co/roberta-large

| N past utt | F1-weighted | F1-macro |
|---|---|---|
| 0 | 0.63773 | 0.46742 |
| 1 | 0.65267 | 0.50598 |
| 2 | 0.65033 | 0.48966 |
| 3 | **0.66317** | **0.50992** |
| 4 | 0.6505 | 0.50185 |
| 1/3 | *0.66638* | 0.*51666* |

Table 3.6: RoBERTa$_{large}$ performance on MELD Test with a different number of past utterances. The last line reports the scores of the model trained using the third and first past utterances as context.

the 2nd past utterance belongs to the current speaker, and their emotional inertia is harder to capture by the model.
With the goal of investigating the phenomenon just mentioned, another setting has been tested. We considered the first and third past utterances since the performances improved back when the 3 past utterances were employed. The results are reported 3.6. Those results indicate a slight improvement with respect to the previous settings. Such results support the initial hypothesis that adding the past dialogue indiscriminately as a context can be detrimental.

**2. Balancing Techniques**

**Experiment Setup** Considering the results obtained in the previous section, we decided to use one past utterance as context. We made this decision because it was the most influential past utterance. This model version was trained with the dataset augmented using resampling, namely upsample. The less populated classes were upsampled, namely *disgust, fear, sadness, anger, surprise*. The upsample proportion was chosen according to their initial sample count to have mostly balanced emotion classes.

- The *disgust* and *sadness* classes were augmented by 1.5 their original size.
- The *fear* class size was doubled.
- The *anger* class was augmented by half its size.
- The *surprise* class was augmented by 0.4 its size.

The final sample count is reported in Table 3.7. We expect the model to benefit from this larger training set and the performances to improve. We also train the model with the *class weight* balancing technique with the weights reported in Table 3.3 and compare the two techniques. Finally, even if the two techniques have similar effects, we tried to combine them and verify if the model improves. In fact, the upsampled training dataset still presents some balance issues and to contrast them the *class weights* introduced in section 3.1.3.2 were used during the training phase. We expected that the performances would improve because the class weights impose a less severe and more uniform penalty on the loss function allowing better performances on the more frequent classes with respect to the weights in Table 3.3.

| Emotion | Count | Class Weight |
|---|---|---|
| neutral | 4710 | 0.3796 |
| anger | 1663 | 1.0752 |
| disgust | 667 | 2.6410 |
| fear | 670 | 2.6687 |
| joy | 1743 | 1.0258 |
| sadness | 1366 | 1.3089 |
| surprise | 1687 | 1.0598 |
| Total | 12516 | - |

Table 3.7: Upsampled MELD train split emotion count and respective class weights

**Results** Table 3.8 reports the results. The model benefits from the upsampled training set and the performances are improved. On the other hand, the model trained using the class weights show signs of overfitting and the model loses some of its generalization capabilities. The expectations on the combined techniques were not met because also this model overfits with respect to the case without class weights. This can be due to the very objective of the class weights. By giving more importance to the less frequent classes, which are the upsampled ones, the model overfits on those examples and loses some of its generalizing capabilities. By inspecting the precision and recall scores of the model trained with and without the class weights and comparing them, it is possible to notice that the recall scores of the *fear* and *disgust* are nearly the same and these are the most augmented classes, whereas it was expected to see an improvement. On the other hand, classes like *sadness* and *anger* experience this improvement. This behaviour might indicate that the class weights combined with upsampling can result in detrimental effects.

| Balance technique | F1-weighted | F1-macro |
|---|---|---|
| No Balancing | 0.65267 | 0.50598 |
| Upsample | **0.6630** | **0.5242** |
| Class Weights | 0.60145 | 0.45681 |
| Upsample and CW | 0.61 | 0.4661 |

Table 3.8: RoBERTa trained with different balancing techniques. The best results are highlighted in bold.

## 3.2 FewShot for ERC

In this section, the chosen method to perform Few-Shot learning is described in detail along with the aspects analyzed during the experiments. Then we define the experimental setups and report the results obtained.

### 3.2.1 Architecture: SetFit

Sentence Transformer Fine Tuning[31] (SetFit) is a prompt-free method proposed to perform few-shot learning in text classification tasks. It employs a sentence transformer[41] to generate meaningful sentence embeddings which are then passed to a classification head. Sentence transformers are modifications of the pre-trained transformer models, such as BERT, which use Siamese and triplet network structures to create a meaningful embedding space where semantically similar sentences are close to each other.
The training defined by the authors of SetFit is done in two distinct phases. Firstly, the sentence transformer is finetuned in a Siamese fashion using contrastive learning, a technique often used in image recognition[42]; By using this strategy it is easier to deal with the limited amount of available data because it allows the generation of a number of triplets to train the model on significantly larger than the initial examples. Formally, given the set $D = \{X, Y\}$ of *K* samples, with *K* small, where $x_i, y_i$ are a *sentence,label* pair, *R* positive triplets are built for each class *c*. These triplets are defined as $T_p^c = \{x_i, x_j, 1\}$, meaning that the two sentence $x_i$ and $x_j$ share the same class. Conversely, negative triplets are built using two sentences that belong to different classes, formally $T_n^c = \{x_i, x_j, 0\}$. The final training set is generated by concatenating all the produced triplets. The number of triplets generated, |T|, follows the following formula:

$$|T| = 2R|C| \tag{3.3}$$

where $|C|$ is the number of samples per class in the given set D, $R$ is a hyperparameter. This procedure allows the generation of a much larger number of training samples with respect to the one initially available. Once the training set is built, the sentence transformer is finetuned using *CosineSimilarity* as loss function, while the classification head's weights are frozen. Its combination with the triplets allows the model to push close to each other sentences that belong to the same class and drive away those that belong to different ones. This procedure allows the model to create a meaningful feature embedding space, which can be exploited to classify the utterances.
The second phase consists of training the classification head. The sentences in the training set $D$ are embedded with the sentence transformer, whose weights are now frozen, creating the input for the head along the labels, namely $T = \{Emb(X), Y\}$. The head is trained using a *CategoricalCrossEntropy* loss. Due to the small amount of data used in this scenario, the model tends to overfit quickly. To limit the effects of this phenomenon, the training is stopped when the validation loss did not improve for a certain number of epochs and the weights were restored to the best loss value.

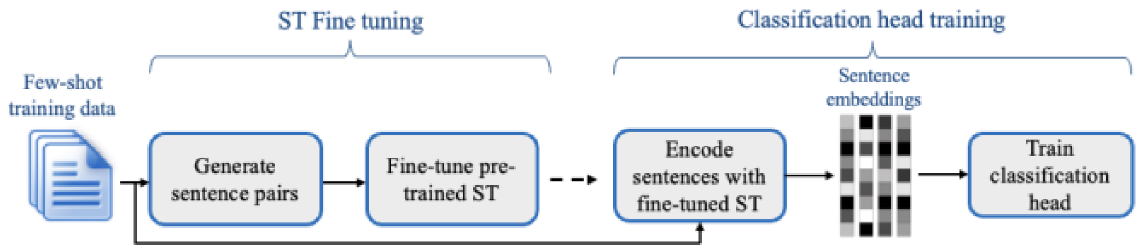

Figure 3.1: SetFit trainig diagram[31].

### 3.2.2 Aspects Analyzed

In the following section, we describe the aspects analyzed of the SetFit method to assess its effectiveness and limitation related to the ERC task.

#### 3.2.2.1 Context Definition in SetFit architecture

The introduction of past utterances in the input as defined in 3.1.3.2 might increase the noise in the embeddings. The addition of the context to the current utterance would increase the data variance significantly. Furthermore, the utterances prepended will not have generally the current utterance's label. Mixing them would decrease the triplets' efficiency because they would contain contrasting information. To address this concern, the experiments in section 3.2.3.1 are carried out.

#### 3.2.2.2 Integration of Contextual information in SetFit

The results that emerged in section 3.2.3.1 indicate that the context definition given in section 3.1.3.2 is not compatible with the ST architecture trained with the SetFit strategy. To address these incompatibilities, the classification head architecture has been modified. The sentence transformer is still used to create the embeddings and is trained with the same method, but the input to the classification head will be both the context and the current utterance's embedding which are merged according to different techniques. The new vector representing the information extracted from both utterances is fed to FC layers to generate the prediction. We hypothesize that adding the context embedding information could give more depth to the current utterance embedding, allowing the model to capture possible nuances that were not evident previously. The experiment in section 3.2.3.2 reports the different merging strategies' performance and their comparison.

#### 3.2.2.3 Performance comparison

To assess the effectiveness of the SetFit method and the modified architecture described in section 3.2.2.2, we need to compare different strategies and other architectures. Section 3.2.3.3 reports the experiments carried out to compare the following elements:

- The few-shot learning and fine-tuning technique.
- The context contribution to the SetFit method.
- How the sentence transformer impacts the performances in the few-shot scenario.

The results should provide insight into how the few-shot learning paradigm, namely the SetFit method with and without context, performs in the ERC task and if it is better than finetuning the model when the available data are limited.

### 3.2.3 Experimental setups and Results

In this section are described the experimental setups and the results obtained.

#### 3.2.3.1 Context Definition in SetFit architecture

**Experimental Setup** As mentioned in section 3.2.2.1, we have to verify if the method as defined can handle the context definition given in section 3.1.3.2. To that end, we trained the model with the same subset of MELD's training split composed of five examples (*shots*) from each class, picked randomly. A different number of past utterances was added as context for each experiment.

**Results** Table 3.9 shows that the performances deteriorate as the number of past utterances increases. These results justify the change in the classification head described in section 3.2.2.2.

| N past utterance | F1-Weighted | F1-Macro |
|---|---|---|
| 0 | 0.30881 | 0.25678 |
| 1 | 0.30356 | 0.21779 |
| 2 | 0.28844 | 0.19504 |
| 3 | 0.22634 | 0.15396 |

Table 3.9: Setfit scores on MELD Test split with past utterances as context prepended to the current sentence.

#### 3.2.3.2 Integration of Contextual information in SetFit

**Experimental Setup** The second experimental scenario aimed to confront different merging strategies and how they impacted the model's performances when trained with a different number of *shots*, namely 5,10,15,20,50,100,150 examples from each class. Each subset of samples is fixed and built incrementally on the smaller ones: the 20-shot subset contains the 15-shot, which in turn will comprise the 10-shot subset, etcetera. The R hyperparameter mentioned in section 3.2.1 was set to 20 for these experiments. Being the classes balanced and remembering Equation 3.3, the number of triplets generated is equal to:

$$|T| = 2 \times 20 \times number\_of\_classes \times n\_shots \tag{3.4}$$

For these experiments, we used the sentence transformer *paraphrase-mpnet-base-v2*[3][41],

[3]https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2

which has 110M parameters. The merging strategies tested are:

1. *Adding* the context embedding vector to the current utterance one, through an Add layer.
2. *Average* of the context and the current utterance embeddings, through an Average layer.
3. *Concatenating* the context and the current utterance embeddings, through a Concatenate layer.

**Results** Figure 3.2 reports the score progression of the model for the chosen shots. It is possible to observe that the *concatenation* performs better than the other two across the whole span of shots. These results can be justified by the fact that the current utterance embedding is not modified and its features carry the bulk of information, while the context should enrich the representation. On the other hand, the *average* and *add* merge strategies alter the current utterance embedding when merging the context embedding, also introducing noise. This difference in the merged vector construction could explain the performance differences between these two strategies and the concatenation one.

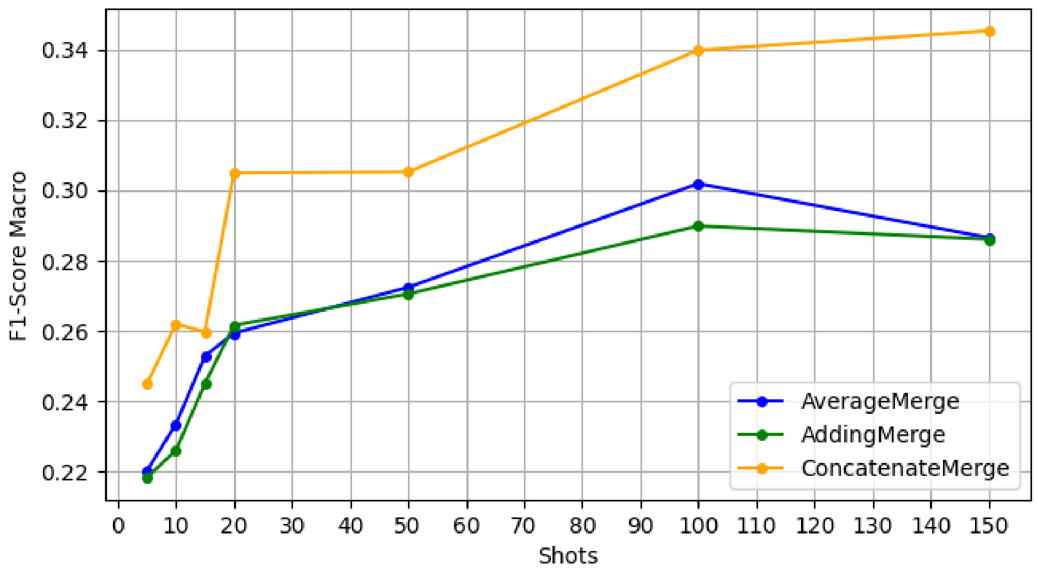

Figure 3.2: Context merge strategies .

#### 3.2.3.3 Performance comparison

**Experimental setup** As mentioned in section 3.2.2.3, the following experiments were performed to gain insight into the methods and architecture effectiveness. To gauge the context contribution to the architecture defined in section 3.2.2.2 it is necessary to test it without it. Therefore, we trained the model without the addition of the context embedding. The sentence transformer *”all-roberta-large-v1”*[4] was chosen for these experiments. The same transformer architecture is used in the experiments in section 3.1 but is finetuned to work as an ST. By choosing a different transformer, we could also gain some insight into how the model dimension affects the performances in this scenario, comparing it with the results presented in section 3.2.3.2 (we called it *ContextSFParaphrase*). We chose concatenation as the merge strategy because it yielded the best results. We tested the following experimental settings:

1. *SetFitRoberta*: the SetFit method tested without the addition of any contextual information, with *all-roberta-large-v1* employed as ST.

2. *ContextSFRoberta*: the concatenation of the context performed with a larger sentence transformer.

3. *Finetuning*: the RoBERTa transformer used in the previous section is finetuned using the same number of shots so that it is possible to confront the two methods’ differences.

**Results** In Figure 3.3 it is possible to observe the evolution of the F1-score macro with respect to the number of shots used.

1. For a small number of shots, the SetFit method performs vastly better than standard finetuning and the dimension of the sentence transformer plays a negligible role in the model’s performance and becomes relevant only for larger amounts of data. The reason for the sharp gap between finetuning and SetFit methods lies in the amount of data used for the training of the model’s body: the former relies only on the given shots, and the latter through triplet generation and contrastive learning actually learns from a much bigger dataset (even if the initial amount of data is the same).

2. It is possible to notice that the context injection to the model as defined in section 3.2.2.2 fails to capture meaningful additional information and the scores are nearly the same for every set of shots.

3. We can observe that the score curve of the few shot learners after a rampant start slows down and almost meets a plateau, meanwhile the finetuning benefits from the larger amount of data and largely improve its performances.

[4]https://huggingface.co/sentence-transformers/all-roberta-large-v1

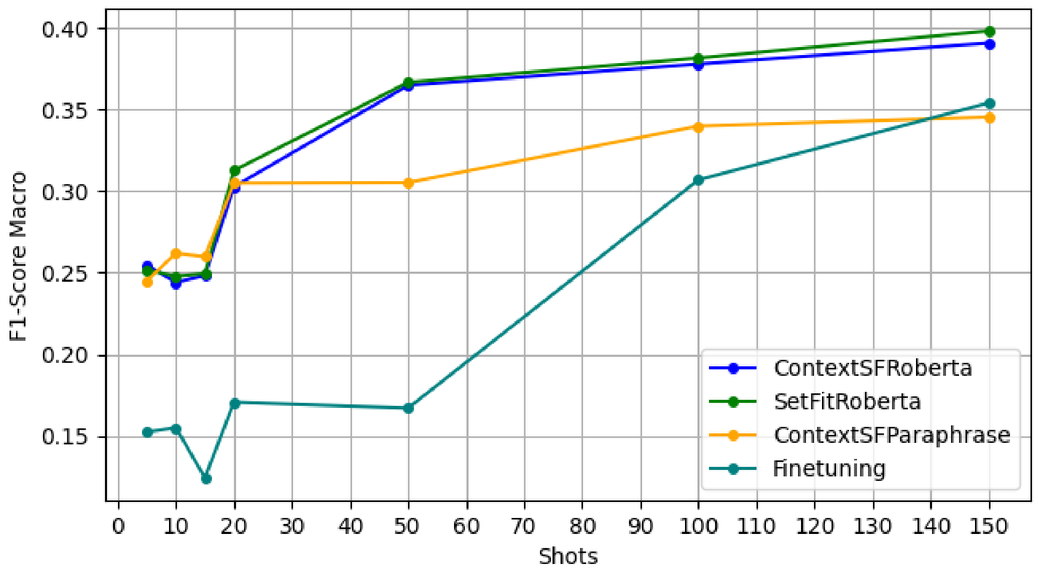

Figure 3.3: Finetune and Fewshot learning performances.

#### 3.2.3.4 Full Dataset Performance Comparison

**Experimental Setup** The following experiments are performed to compare how the two training methods, namely Finetuning and SetFit, fare when the whole dataset is available. We kept the context fixed to one past utterance. As far as the finetuning is concerned, the results used are the ones presented in section 3.1 for the complete dataset and those in the previous section for the low shots values. The same goes for the few-shot learning scores, we only train the model with "all-roberta-large-v1" using all the samples available in MELD's training split. For this experiment, the R hyperparameter was tested in a range between 2 and 5 to find the optimal value. We kept the range small to keep the training time limited. The R hyperparameter is then set to 3 because the performances started to decline after it.

**Results** Figure 3.4 reports the score progression of the to models. We can be observed the finetuned model outperforms the model trained with the SetFit method. The reason behind this difference could lie in the context information, which *RoBERTa* integrates better than the *ContextSFRoberta* model, and in the nature of the representation generated by the two encoders.

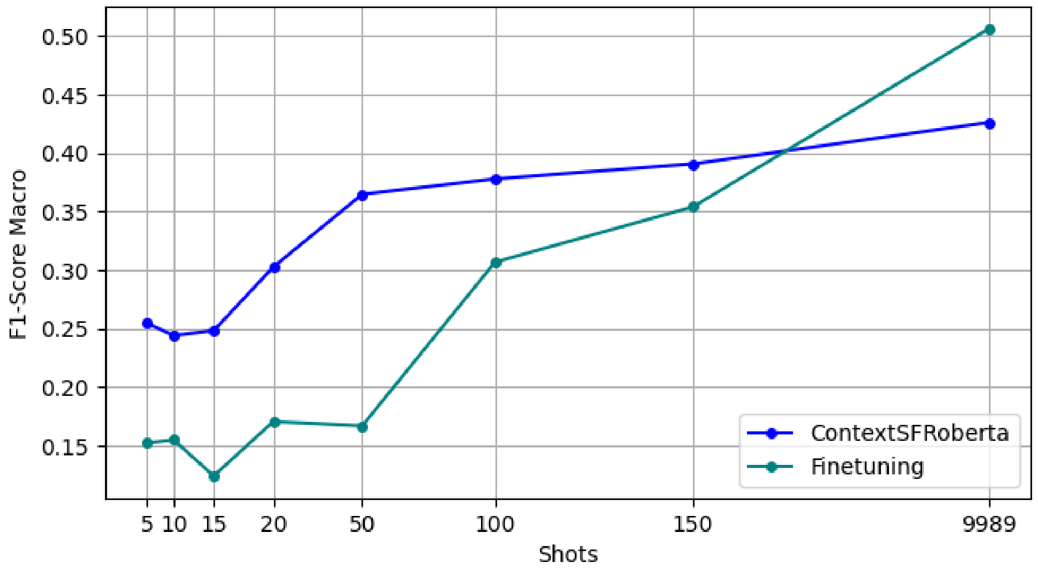

Figure 3.4: Models' trained using the Finetune and Fewshot learning techniques with MELD's full training split.[riscrivere]

### 3.2.4 Interdataset performances of finetuning and FSL

The last experiments aimed to understand if the few-shot learning paradigm could be more effective than using Transfer Learning or fine-tuning models with few samples. This method could be helpful in scenarios where an already trained model has to adapt to different data whose availability is limited.

**Experimental Setup** In the fine-tuning case, we trained the whole model. Conversely, when using Transfer Learning, the encoder block of the models, namely RoBERTa and the sentence transformer *all-roberta-large-v1*, are frozen during training and only the classification head's weights are updated. We used a subset of data extracted from DailyDialog's training set to simulate the limited data scenarios and train the models with said methods on an increasing number of shots, namely 0,5,10,15,20,50,100,150 examples per class. We used the models trained with the whole MELD's training split in the previous section as a starting point. The performances are then evaluated on the test sets of both MELD and DailyDialog. The performance differences should provide insight into the methods' effectiveness.

**Results** Figure 3.5 and Figure 3.6 report the evolution of the macro F1-score of the SetFit architecture and RoBERTa architecture, respectively. The performances generally decrease with

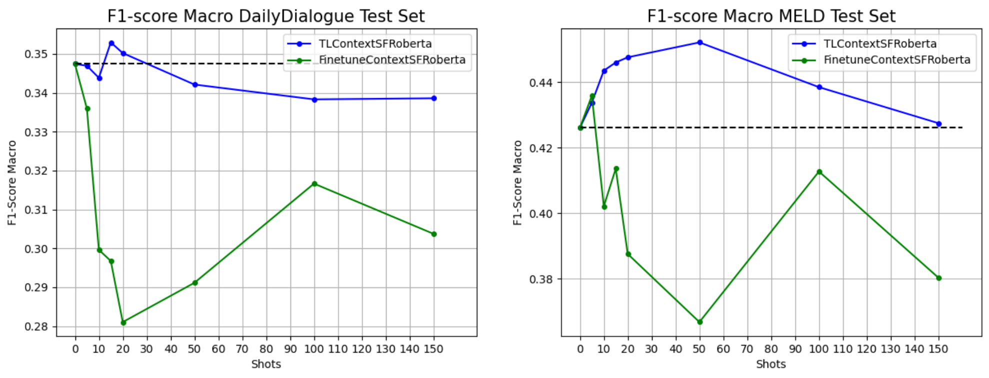

Figure 3.5: Transfer Learning and Finetuning on few-shots from DailyDialog's training split of *ContextSFRoBERTa*, previously trained with the SetFit strategy on MELD's training split. The black line represents the model's performance without additional training (0-shot).

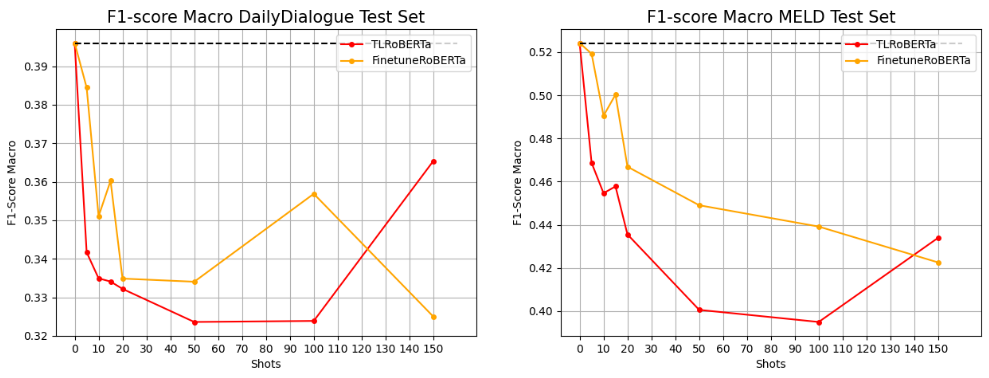

Figure 3.6: Transfer Learning and Finetuning on few-shots from DailyDialog's training split of *RoBERTa* previously fine-tuned on MELD's training split. The black line represents the model's performance without additional training (0-shot).

respect to the model not trained on DailyDialog's subsets and show unstable behaviour. The SetFit model trained using Transfer learning (*blue curve* in Figure 3.5) shows an improvement with a low number of shots on MELD's test set, peaking at 50 shots. In Figure 3.6 the transfer learning performances (*red curve*) degrade rapidly but show signs of improvement with a larger amount of data. Both models' performance decreased when further fine-tuned (*green and gold curves*) and showed irregular trends. The models overfit heavily on the small subsets which would explain the performance deterioration, but the results obtained are not enough to draw substantial conclusions and further experiments would be necessary. Nevertheless, the transfer learning appears to be more stable than finetuning of the models. Furthermore, The transfer learning trained SetFit model's curve suggests that it can handle the limited amount of data better than its RoBERTa counterpart. This might indicate that the sentence embeddings yield a more flexible representation with respect to the one produced by the RoBERTa architecture. As far as the finetuning is concerned, the performance degradation could also derive from the noise introduced in the encoded representation by the differences in the utterances' text structure and content. The differences in the annotation process of the two datasets should play a minor role, if any, in this instance otherwise we would expect the performances on DailyDialog to improve and those on MELD to decrease, but the curves generally follow the same trend. Again, to gather more insight and find support for these hypotheses further experiments should be carried out.

# Chapter 4

# Discussion

In this section, we discuss the challenges encountered and the criticalities that emerged during the experimental phase and possible margins of improvement.

## 4.1 Emotion Recognition in Conversation

The architectures used in this experimentation are taken from the literature and tested with different scenarios to gain insight into different aspects of the task.

#### Limitations

The lack of real data makes difficult the evaluation of the model performance in the wild. The data encountered might significantly differ from the training data and also present new challenges, such as a different way to express emotion according to the domain of application, and differences in the language used, such as vocabulary and sentence articulation, and noisier text, presenting typos or grammatical errors. The RoBERTa model utilizes a static context definition which improves the performance, but it is limited by its lack of flexibility. Furthermore, no information about the speakers or their emotional dynamics has been integrated into the proposed model.

#### Future Developments

Possible improvements in this aspects are:

- *Real Data*: gather real data to evaluate the model's behaviour on it. The Adversarial Augmentation techniques mentioned in section 3.1.3.2 could be explored in this phase if it is appropriate with respect to the data's nature.

- *Improve Context Definition*: in light of the results of the context experiments carried out in section 3.1.4.2, a possible improvement is to define a more complex decision process to pick the contextual past utterances rather than use a fixed number. In this way, the noise introduced by the contextual utterance could be reduced and the information gained increase.
- *Test different architectures*: other architectures from the literature could be tested and merged together to integrate different aspects of the task, such as CRFs for Emotion Dynamics and LSTMs for speakers' information.

## 4.2 Few-Shot Learning for ERC

As far as Few-Shot Learning applied to ERC is concerned, the experiments prove that methods such as SetFit can provide a strategy to train models with limited data, but the model implemented fails to capture characteristic aspects of the task.

**Limitations** The main challenges arisen in this experimentation are:

- *Architectural limitations*: the employment of a sentence transformer to generate meaningful embeddings by the SetFit method imposes certain limitations in the ERC task. As mentioned in section 1.1, in this scenario identical utterances may convey very different emotions, but the embeddings of these utterances do not reflect this difference. The integration of contextual information in the model to reduce the impact of this issue and generally enhance the representation was not successful.
- *Overfitting*: the small amount of data used in the few-shot experiments leads to overfitting the models. Even if in the experiments reported in 3.2.3, we were able to partially contain this phenomenon, it was not the case in section 3.2.4. This phenomenon might depend on several factors:
  - The architecture is too complex for the given task and other
  - The training parameters are not optimal for this scenario.
  - The regularization techniques employed are not appropriate.

- *Data used*: the subsets used to train the models in the Few-Shot learning experiments were balanced, whereas the test sets of the dataset have an unbalanced label distribution. This could influence the evaluation yielding a slightly pessimistic estimation.

The experiments to adapt the trained models to new data require further investigation. While transfer learning applied to the architecture proposed to tackle the FSL showed better performances with respect to the other scenario, it is not enough to draw conclusive results. The hyperparameter could not be optimal for the other settings, mainly the R parameter that determines the number of triplets generated could influence the quality of the

**Future developments**

There are several areas where improvements could potentially be made in the architecture used and in the experimental setups:

- *Context Integration*: develop a method to integrate the context information in an architecture as it is defined in the SetFit strategy. Using an LSTM to keep track of the dialogue evolution could be a starting point.
- *Cross-validation*: to remove the bias introduced by using the same subsets in the Few-Shot experimentation, the experiments should be repeated with different fixed subsets in multiple runs and average the results. This could yield also a measure of the variability of the performances derived from the training data. It should be noticed that the limited amount of examples in certain classes could cause the subsets to overlap for larger shots number.
- *Hyperparameters search*: the hyperparameters used in section 3.2.3 may not be the optimal ones and searching for better configurations could enhance the performance.
- *Regularization techniques*: to try to reduce the model overfitting different regularization techniques should be explored, such as regularizers.

# Chapter 5

# Conclusions

In this work, we explored methods and architectures used in Emotion Recognition in Conversation and Few-Sot Learning applied to this task. The scope of this thesis was to explore some of the challenging aspects of the ERC task, such as context modelling and data unbalance, with a reactive chatbot application as a reference case study. These aspects have been explored in the first phase of the thesis. The lack of real data to train the model on, the difficulty of gathering such data, and the expensive annotation process inspired the employment of Few-Shot learning techniques; which have been explored in the second phase of the study.

In the first part of the study, a few baselines were tested to define the performance starting point and, among them, the Text-CNN was used to test different preprocessing pipelines to gather insight into the importance of the textual elements. These experiments revealed that punctuation marks provide key information to the model, significantly enhancing the performances on MELD's test set.

The RoBERTa architecture along with the context definition defined by [20], described in section 3.1.3.2, was used to explore a different number of utterances used as context and gauge their contribution to the performances. The introduction of the preceding utterance in the dialogue proved to be beneficial for the model, but adding more elements of the past dialogue as context introduces noise, limiting the improvement. Preliminary tests on a different context structure suggest that a choice of the most relevant past utterances tailored to the current one could lead to better results. After fixing a context configuration, several balancing techniques, namely class weights and upsampling, are explored to deal with the unbalanced label distribution reported in many datasets related to the task. The class weights proved to be less effective than upsampling in the tested setting.

The second phase of the study comprises the exploration of the Few-Shot learning techniques involved testing the SetFit method and applying it to the ERC task. The architecture trained with this strategy was incompatible with the given context definition and required to be adapted to support the integration of contextual information. By inspecting the results, the proposed modification of the classification head failed to capture the contextual information. The introduction of an LSTM layer in this part of the model to keep track of the dialogue evolution could be a starting point for future developments to integrate context modelling. Nevertheless, this method proved to be more effective than fine-tuning in training the model with limited data. As far as the use of the SetFit method as a more efficient strategy than fine-tuning to train on a limited amount of data, starting from a model trained beforehand on other data related to the task, the results were inconclusive. Other than formulating several hypotheses to explain the models' behaviour, reported in section 3.2.4, we cannot draw any solid conclusions. Further investigations are required to obtain conclusive evidence about this topic.
This work provides insight into some aspects of both ERC and FSL applied to ERC and sets a starting point for future development.

# Bibliography

[1] Ahlam Alnefaie et al. “An Overview of Conversational Agent: Applications, Challenges and Future Directions”. In: *WEBIST*. SCITEPRESS, 2021, pp. 388–396.

[2] Rosalind W Picard et al. “Affective computing, 1997”. In: *Google Scholar Google Scholar Digital Library Digital Library* (1997).

[3] Patrı́cia Pereira, Helena Moniz, and João Paulo Carvalho. “Deep Emotion Recognition in Textual Conversations: A Survey”. In: *CoRR* abs/2211.09172 (2022).

[4] Mallikarjuna Rao Gundavarapu et al. “Empathic Chatbot: Emotional Astuteness for Mental Health Well-Being”. In: *Applied Computational Technologies*. Ed. by Brijesh Iyer, Tom Crick, and Sheng-Lung Peng. Singapore: Springer Nature Singapore, 2022, pp. 697–704. ISBN: 978-981-19-2719-5.

[5] Sarada Devaram. *Empathic Chatbot: Emotional Intelligence for Empathic Chatbot: Emotional Intelligence for Mental Health Well-being*. 2020. DOI: 10.48550/ARXIV.2012.09130. URL: https://arxiv.org/abs/2012.09130.

[6] Soujanya Poria et al. “MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations”. In: *ACL (1)*. Association for Computational Linguistics, 2019, pp. 527–536.

[7] Yanran Li et al. “DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset”. In: *IJCNLP(1)*. Asian Federation of Natural Language Processing, 2017, pp. 986–995.

[8] Patrı́cia Pereira, Helena Moniz, and João Paulo Carvalho. “Deep Emotion Recognition in Textual Conversations: A Survey”. In: *CoRR* abs/2211.09172 (2022).

[9] Soujanya Poria et al. “Emotion Recognition in Conversation: Research Challenges, Datasets, and Recent Advances”. In: *IEEE Access* 7 (2019), pp. 100943–100953.

[10] Paul Ekman. “Basic Emotions”. In: *Handbook of Cognition and Emotion*. John Wiley & Sons, Ltd, 1999. Chap. 3, pp. 45–60. ISBN: 9780470013496.

[11] Robert Plutchik. “A psychoevolutionary theory of emotions”. In: *Social Science Information* 21.4-5 (1982), pp. 529–553.

[12] Albert Mehrabian. “Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in Temperament”. In: *Current Psychology* 14 (1996), pp. 261–292.

[13] Michael W. Morris and Dacher Keltner. “How Emotions Work: The Social Functions of Emotional Expression in Negotiations”. In: *Research in Organizational Behavior* 22 (2000), pp. 1–50. ISSN: 0191-3085. DOI: https : / / doi . org / 10 . 1016 / S0191 - 3085(00 ) 22002 - 9. URL: https : / / www . sciencedirect . com / science / article / pii / S0191308500220029.

[14] Carlos Busso et al. “IEMOCAP: interactive emotional dyadic motion capture database”. In: *Lang. Resour. Evaluation* 42.4 (2008), pp. 335–359.

[15] Chao-Chun Hsu et al. “EmotionLines: An Emotion Corpus of Multi-Party Conversations”. In: *LREC*. European Language Resources Association (ELRA), 2018.

[16] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://aclanthology.org/D14-1181.

[17] Navonil Majumder et al. “DialogueRNN: An Attentive RNN for Emotion Detection in Conversations”. In: *AAAI*. AAAI Press, 2019, pp. 6818–6825.

[18] Deepanway Ghosal et al. “DialogueGCN: A Graph Convolutional Neural Network for Emotion Recognition in Conversation”. In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 154–164.

[19] Zaijing Li et al. “EmoCaps: Emotion Capsule based Model for Conversational Emotion Recognition”. In: *ACL (Findings)*. Association for Computational Linguistics, 2022, pp. 1610–1618.

[20] Taewoon Kim and Piek Vossen. “EmoBERTa: Speaker-Aware Emotion Recognition in Conversation with RoBERTa”. In: *CoRR* abs/2108.12009 (2021).

[21] He Wang et al. "M2FNet: Multi-granularity Feature Fusion Network for Medical Visual Question Answering". In: *PRICAI (2)*. Vol. 13630. Lecture Notes in Computer Science. Springer, 2022, pp. 141–154.

[22] Xiaohui Song et al. "Emotionflow: Capture the Dialogue Level Emotion Transitions". In: *ICASSP*. IEEE, 2022, pp. 8542–8546.

[23] Yaqing Wang et al. "Generalizing from a Few Examples: A Survey on Few-shot Learning". In: *ACM Comput. Surv.* 53.3 (2021), 63:1–63:34.

[24] Archit Parnami and Minwoo Lee. "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning". In: *CoRR* abs/2203.04291 (2022).

[25] Oriol Vinyals et al. "Matching Networks for One Shot Learning". In: *NIPS*. 2016, pp. 3630–3638.

[26] Jane Bromley et al. "Signature Verification Using a Siamese Time Delay Neural Network". In: *NIPS*. Morgan Kaufmann, 1993, pp. 737–744.

[27] Jake Snell, Kevin Swersky, and Richard S. Zemel. "Prototypical Networks for Few-shot Learning". In: *NIPS*. 2017, pp. 4077–4087.

[28] Lisa Torrey and Jude Shavlik. "Transfer learning". In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[29] Gaël Guibon et al. "Few-Shot Emotion Recognition in Conversation with Sequential Prototypical Networks". In: *EMNLP (1)*. Association for Computational Linguistics, 2021, pp. 6858–6870.

[30] Sachin Ravi and Hugo Larochelle. "Optimization as a Model for Few-Shot Learning". In: *ICLR*. OpenReview.net, 2017.

[31] Lewis Tunstall et al. "Efficient Few-Shot Learning Without Prompts". In: *CoRR* abs/2209.11055 (2022).

[32] Sayyed M. Zahiri and Jinho D. Choi. "Emotion Detection on TV Show Transcripts with Sequence-Based Convolutional Neural Networks". In: *AAAI Workshops*. Vol. WS-18. AAAI Technical Report. AAAI Press, 2018, pp. 44–52.

[33] Ankush Chatterjee et al. "SemEval-2019 Task 3: EmoContext Contextual Emotion Detection in Text". In: *SemEval@NAACL-HLT*. Association for Computational Linguistics, 2019, pp. 39–48.

[34] Saif M. Mohammad and Peter D. Turney. "Crowdsourcing a Word-Emotion Association Lexicon". In: *Comput. Intell.* 29.3 (2013), pp. 436–465.

[35] Dou Hu, Lingwei Wei, and Xiaoyong Huai. "DialogueCRN: Contextual Reasoning Networks for Emotion Recognition in Conversations". In: *ACL/IJCNLP (1)*. Association for Computational Linguistics, 2021, pp. 7042–7052.

[36] Yan Wang et al. "Contextualized Emotion Recognition in Conversation as Sequence Tagging". In: *SIGdial*. Association for Computational Linguistics, 2020, pp. 186–195.

[37] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation". In: *EMNLP*. ACL, 2014, pp. 1532–1543.

[38] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019).

[39] Ashish Vaswani et al. "Attention is All you Need". In: *NIPS*. 2017, pp. 5998–6008.

[40] John X. Morris et al. "TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP". In: *EMNLP (Demos)*. Association for Computational Linguistics, 2020, pp. 119–126.

[41] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 2019, pp. 3980–3990.

[42] Gregory R. Koch. "Siamese Neural Networks for One-Shot Image Recognition". In: 2015.