NLP - Standard Project Report
NLP Course Project

Barnabé Sellier, Valentin Koenig, Maxence Murat and **Benoit Noemie-Malakin**
Master's Degree in Artificial Intelligence, University of Bologna
{ barnabe.sellier, valentin.koenig, maxence.murat, benoit.noemie }@studio.unibo.it

Abstract

The Standart Course Project of NLP consist of solving subtask iii of the 10th task in SemEval 2024. To do so, we will first implement an ERC model that classify emotions withing a dialog and then an EFR model in order to predict triggers of an emotion-flip. We solve those task using a fine-tuned BERT model and we find that the Full BERT has significant better result than the Freezed one.

1 Introduction

SemEval is a competition of natural language processing (NLP) tasks that takes place every year and whose goal is to advance state of the art in semantic analysis. In this paper we will be answering to the 10th task of the 2024 edition which is *Emotion Discovery and Reasoning its Flip in Conversation* (EDiReF) and more specifically to the sub-task iii: Emotion Flip Reasoning (EFR) in English conversations. Given a dialogue EFR aims to identify the trigger utterance(s) for an emotion-flip in a multi-party conversation dialogue. In order to solve this problem we will also do the Emotion Recognition in Conversation (ERC) task which consist of predicting the emotion associated to an utterance. Where past application of emotion recognition were mostly focused on standalone text, such as review for e-commerce, ERC goes further as emotion recognition is made for a whole conversation, making it possible to analyse evolution of emotions along an exchange. The next step is to find causes of the emotion flip, which is the goal of EFR. The combination of ERC and EFR can show itself very useful for some applications such as dialogue agents as they can tell information about the mental state of the user and what response made him in this state. Thanks to that, the dialogue agent can be improved by taking an emotion positive answer of the user as a reward and a negative one as a penalty, and then adapt itself to provide the most relevant answer in the given context. EFR task has been first introduced in (Kumar et al., 2021), along with the MELD-FR dataset, a specifically augmented MELD (Poria et al., 2019) dataset for this task. They use GRUs and a Masked Memory Network for ERC and then transformer encoders for EFR. (Kumar et al., 2023) introduced TGIF a model, using both GRUs and Transformers for the EFR task, whose goal is to identify the nature of the triggers. Here we will build a model from BERT to tackle both ERC and EFR task, we will train and test the models on the MELD-FR dataset.

2 Background

The EFR task is the very novel one in the field of NLP, the first paper to introduce it, (Kumar et al., 2021), is from 2021 and has been few explored since then, as the only other paper about this task if from the same team and has the goal to find the nature of the triggers in addition to detect them. The first one achieved a F1-score of 53.9% by using a Transformer-based network. The second one introduced TGIF, using GRUs along with Transfomers and obtain an F1-score of 38.8% for the task of classifying the kind the of triggers. For the ERC task, several papers and models has been made on the subject. EmoBERTa (Kim and Vossen, 2021) uses RoBERTa, and takes advantage of the speaker information to have a F1-score of 65.61%. BERT-ERC (Qin et al., 2023) shows that fine-tuning BERT is enough for this task and use a two-stage training, teacher and student, to find a F1-score of 67.11% on the MELD dataset.

3 System description

In order to produce results for the third subtask we have to create two models, one for the ERC task, focusing on the emotion recognition in conversation, and a second one to detect the emotion flip in the EFR task. To build a sentiment analysis model we need to feed it a training dataset composed of words or, in our case, sentences and their emotion labels. However to make a model capable of detecting emotion flips and their triggers in a dialogue, we need more context information about the dialogue. Both models use the *bert-base-uncased* BERT pre-trained model from Huggingface as a base and a Multi-Layer Perceptron as the classification head, the archicture of both models is shown on Figure 1. The main difference between them is the input format and the output size. For the ERC we just pass the tokenized utterance to the BERT and the output is a 7 sized tensor corresponding to the score of each emotion. The EFR model is more complex as its need global information about the dialogue in order to determine which utterance was the trigger of the emotion-flip. This is why the input is composed of both utterance and emotion and we indicate if it's a trigger. The input format is an array of all the part of dialogue with the following format: u_i e_i $TRIGGER$ if the utterance is a trigger, u_i e_i otherwise, with u_i and e_i corresponding respectively to the i^{th} utterance and i^{th} emotion in the dialogue. This allows the mode to have information about past future utterances and their emotion which very important to find the trigger.

To find the best models for our tasks, we trained them on five different seeds. We ran the same pipeline over every seeds, which is composed of a training phase, a validation phase and then a testing phase with the computation of the F1-scores in order to determine which model we should use. This pipeline is used both on the ERC and EFR model. To build our ERC model we took inspiration from the model of Praveen, which he made available on the Kaggle website. He uses it to predict one of six emotions, so it is particularly well fitted to our needs. For the EFR, we designed it on our own. Overall, our models are implementations of BERT that we specialised for the given tasks.

4 Data

We conducted our experiments with the MELD-FR dataset, an augmented version of the Multimodal EmotionLines Dataset (MELD), adding a trigger label (0 or 1) for each utterance in a dialogue. The MELD dataset is composed of around 13,000 utterances from 1,433 dialogues extracted from the TV-series *Friends* where each utterance is labeled an emotion based on Ekman's six universal emotions (Joy, Sadness, Fear, Anger, Surprise, and Disgust) and an additional Neutral label if no emotion is present. The dataset is provided by SemVal and available on their website in the page dedicated to the 10th task accessible by the url given in Links to external resources. As a dialogue can have several emotion-flip, they are splitted in a way that each row represent only one emotion-flip. This is why we end up with 4,000 different rows, some being a subset of the following one. Then a single emotion flip is composed of either one or several triggers.

4.1 Some Statistics

The dataset is composed of several emotions and speakers. We find it interesting to extract some statistics from the data as it could be helpful for the error analysis. In Table 1 we can see that there is a majority of *neutral*, and that *disgust* and *fear* appear the least.

Table 1: Emotion count in the dataset.

anger	disgust	fear	joy	neutral	sadness	surprise
3964	1049	1114	6317	15263	2648	4645
11%	3%	3%	18%	44%	8%	13%

In term of speakers, there are 231 of them, the majority appearing a few times - the median of the number of appearance of a speaker is 12. As we can expect, the most present speakers are the main characters of the series *Friends*, respectively *Joey*, *Ross*, *Rachel*, *Monica*, *Chandler* and *Phoebe*.

4.2 Pre-processing

Before doing any modification in the dataset it is important to check for missing values. Indeed, in the provided dataset, some NaN values were given in place of trigger values. The first step was to identify them and replace them with a 0 value, i.e. a non-trigger label. Once the cleaning is done we can start to prepare the dataset for training and inference; i.e. we create an other column which is a combination of utterance and its emotion that will be tokenized and then pass as the input of the model. For the ERC task, we just extracted all of the utterances from each dialogue and flatten it to 1 dimension array, we did the same for the emotions.

4.3 Data split

In our first experiments we had applied a random split with shuffle in order to split our dataset in a training, validation and testing set. We then noticed that this method induced a data leakage in our

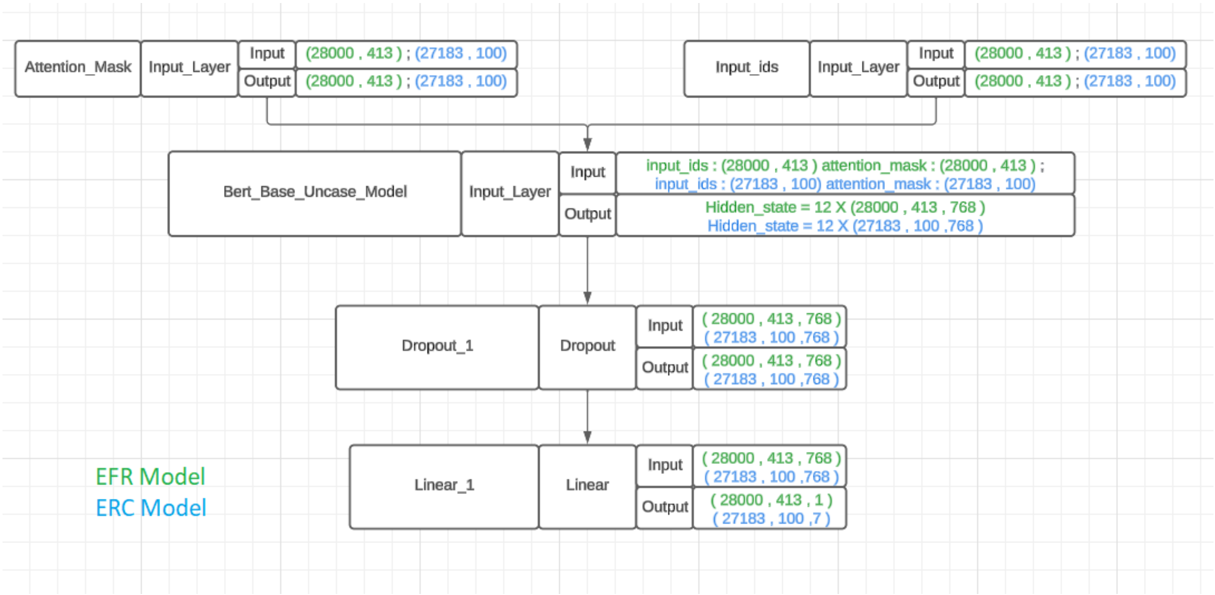

Figure 1: Architecture of the Models.

model as same parts of a dialogue are present several time in the dataset. Thus the model could have dialogue used in the training phase and also find a subset of this dialogue in the test phase which could falsify the results and the real efficiency of the model. To fix this we separated our dataset in a sequential way to avoid same parts of a dialogue being present both in the training and the test set. We end use 80% of the dataset for training and 10% for both validation and testing.

5 Experimental setup and results

Our experimental setup is composed of the pipeline described in the part 3. We trained both ERC and EFR on five seeds, and use the validation phase to determine which model to use based on his score. In order to have a baseline to compare our results we use two Dummy Classifiers, one with the Most frequent strategy and another with the Uniform strategy, as they serve as a simple baseline to compare against other more complex models. In a first place, we implemented TinyBERT, a variant of BERT, in order to gain computation time during the training phase and be able to tweak our model quicker. Indeed, it offers a streamlined architecture tailored for scenarios where computational resources are limited or where rapid inference is paramount. By virtue of its reduced size and parameter count, TinyBERT improves computational efficiency, making it particularly well-suited for deployment in resource-constrained environments or on devices with limited processing capabilities. We then used the two architectures described below, as they obtained more precise results. Also, we mainly used the F1-score metric to assess our models, though we also evaluated our validation set through an accuracy score.

5.1 ERC Model Set up and Results

We first tokenize our data with a max length of 100 as it fits the larger utterance in our dataset. We also don't take into account the fact that some utterance are duplicated in our training phase as it is proven to improve results on Large Language Models (LLM) (Lee et al., 2022). Indeed, as we flatten the utterances of all dialogues in a one dimensional array, some repeat themselves several times. After that, inputs are encoded by BERT that release an output of size 768, we then use a MLP classification head composed of a dropout layer with $p = 0.3$ and a linear layer that outputs seven scores, one for each emotion. For the training of the ERC model, we used a learning rate of 2e-5, a batch size of 32 and 3 epochs as hyper-parameters. We also tested a batch size of 16, but we got worse results that can be explained by the fact that increasing batch size bring a better generalization. We obtain the macro F1-scores of emotion prediction for each seed on Table 2.

5.2 EFR Model Set up and Results

At the core of the architecture lies the BertModel, which serves as the primary encoder responsible

Table 2: F1-score results for the ERC model.

(a) F1-score per seed

Seed	38921	101	27	2839	12
F1-score	0.37	0.43	0.41	0.41	0.41

(b) Mean and Standart Deviation

Mean	Standart Deviation
0.40	0.02

for capturing contextualized representations of input sequences. We instantiated it using the "bert-base-uncased" pretrained model, a widely used variant of BERT with 12 transformer layers and 768-dimensional hidden states. Leveraging the bidirectional nature of transformers, the BertModel can effectively capture contextual information from both preceding and succeeding tokens within the input sequence, facilitating robust feature extraction for downstream tasks.

We then incorporated a dropout layer into the architecture. Dropout regularization is applied to mitigate over-fitting by randomly deactivating a proportion of input units during training. Here, a dropout probability of 0.3 is employed, indicating that 30% of input units are randomly dropped during each training iteration. This regularization mechanism enhances the generalization capability of the model by preventing it from relying too heavily on specific features.

Lastly, we added a linear layer to the architecture, serving as the final classification layer. This linear layer projects the high-dimensional output from the BERT encoder to a single dimension, facilitating binary classification for the emotion prediction task. With an input size of 768 (corresponding to the dimensionality of the BERT hidden states) and an output size of 1, the linear layer produces a probability score representing the likelihood of the positive class (e.g., presence of a particular emotion) for each input sequence.

For the hyper parameters, we determined the batch size on 8 for both training and validation phases, dictating the number of samples processed concurrently in each training and validation iteration. This choice strikes a balance between computational resource utilization and model convergence, facilitating stable training dynamics. Subsequently, the number of training epochs was deliberated upon. Our model underwent training for 3 epochs, allowing it to iteratively learn from the entire training dataset multiple times. This decision was reached to ensure sufficient exposure to the data while mitigating the risk of over-fitting. The learning rate was set to 1e-05 (0.00001). This choice was made judiciously to facilitate stable convergence during training, enabling the model to efficiently navigate the parameter space while minimizing the risk of divergence. We also used the AdamW optimizer, a variant of the original one, with the difference lying in its handling of weight decay regularization. AdamW inherits the core principles of Adam, utilizing adaptive learning rates and momentum to optimize the model parameters. It incorporates the weight decay directly into the optimization step, unlike traditional Adam, which applies weight decay to the parameter updates separately. This modification has been shown to yield more stable and reliable training dynamics, especially for models with large parameter spaces.

Table 3: F1-score results for the EFR model.

(a) F1-score per seed for Full BERT for Validation (V) and Test (T).

Seed	123	246	369	12	24	Mean	SD
F1-score (V)	0.79	0.78	0.79	0.77	0.78	0.78	0.01
Unrolled F1-score (V)	0.79	0.78	0.79	0.78	0.78	0.78	0.005
F1-score (T)	0.72	0.73	0.73	0.71	0.70	0.72	0.01
Unrolled F1-score (T)	0.75	0.74	0.74	0.72	0.72	0.73	0.01

(b) F1-score per seed for Freezed Bert for Validation (V) and Test (T).

Seed	123	246	369	12	24	Mean	SD
F1-score (V)	0.59	0.59	0.60	0.59	0.59	0.59	0.004
Unrolled F1-score (V)	0.56	0.57	0.59	0.58	0.58	0.58	0.01
F1-score (T)	0.54	0.55	0.54	0.55	0.55	0.55	0.005
Unrolled F1-score (T)	0.53	0.54	0.54	0.54	0.55	0.54	0.006

6 Discussion

The ERC model shows poorer results than the EFR model. This could be explained by the lack of context and the larger number of classes to predict. Moreover the classes are not well balanced in the dataset as we can see on Table 1, *neutral* and *joy* are much present than *fear* and *disgust*, which is reflected in the results. Indeed we can see on the Confusion Matrix, Figure 2, that the most well classified emotion are also the more present. Moreover we can see that most misclassified predictions are misclassified as *neutral* or *joy*, e.g. *joy* as *neutral*

(176), *sadness* as *neutral* (110) and *anger* as *neutral* (108). We can also notice that *joy* is a lot misclassified as *neutral* event tough it's very present in the dataset.

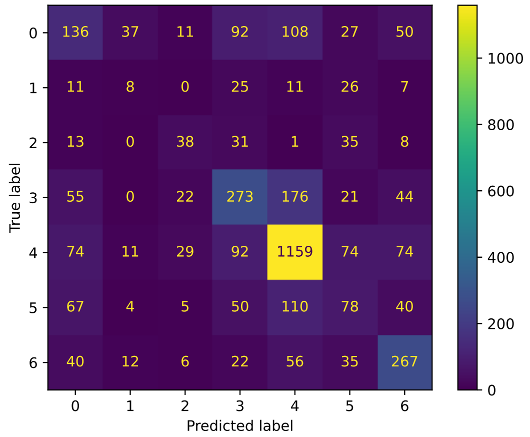

Figure 2: ERC Confusion Matrix.

For the EFR model, the results obtained from the evaluation of the BERT models on the test set reveal interesting insights of their performance. Firstly, the full BERT model consistently outperforms the freeze BERT model across all evaluation metrics. This indicates that fine-tuning the BERT base-uncased model leads to better results compared to using a frozen version of the same model. The average sequence F1 score for the full BERT model is notably higher than that of the freeze BERT model, suggesting that allowing the BERT model to update its parameters during training improves its ability to capture complex patterns in the data.

Comparing the performance of the BERT models with baseline classifiers, such as dummy classifiers, further highlights their effectiveness. The sequence F1-scores obtained by both BERT models are substantially higher than those of the dummy classifiers, indicating that the BERT models have learned meaningful representations from the input data and can accurately classify sequences. Additionally, the unrolled sequence F1-scores of the BERT models are consistently higher than the sequence F1-scores, implying that considering the entire sequence rather than individual tokens leads to better classification performance. The ROC curve plot is shown on Figure 3, comparing the False positive rate with respect to the True positive one. The Area Under the Curve (AUC) is 0.73, which shows a good ratio of true positives in comparaison to false positives.

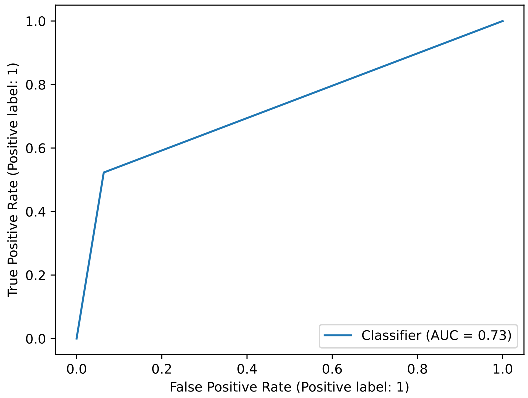

Figure 3: EFR ROC Curve.

One error observed in both BERT models is the misclassification of sequences containing ambiguous or context-dependent tokens. For example, sequences with sarcasm or irony may be misclassified due to the nuanced understanding required to accurately interpret such language cues. Additionally, sequences with out-of-vocabulary (OOV) tokens or rare words may pose challenges for the models, leading to errors in classification. Here is the Confusion Matrix plot showing the 'True' and the 'Predicted' labels for the Freeze Bert model.

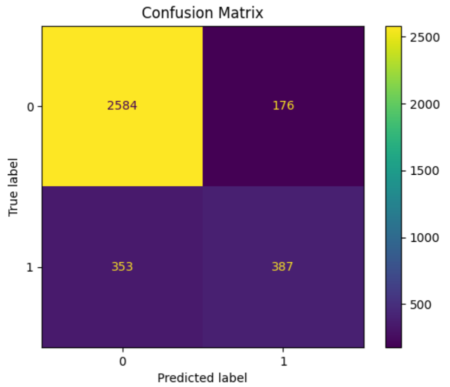

Figure 4: EFR Confustion Matrix.

To address these issues, future developments of this work could focus on incorporating domain-specific knowledge or additional contextual information to enhance the model's understanding of complex linguistic phenomena. Additionally, exploring techniques such as data augmentation or ensemble learning could help improve the robustness and generalization capabilities of the models, potentially reducing errors in classification.

Another limitation worth to mention but not directly related to the task is the lack of GPU space encountered on Google Colab during the training of our models. Google Colab provides users with free access to a NVIDIA T4 GPU. This GPU has finite memory capacities and computational capabilities, which may not be sufficient for intensive deep learning tasks, especially when dealing with large models like BERT and extensive datasets. Furthermore, each session has a maximum runtime limit, typically 12 hours, after which the session is terminated, and all resources, including GPU instances, are released. If model training exceeds this time limit, the session may be terminated prematurely, causing interruptions in training and potentially leading to GPU space constraints upon session restart. To mitigate the impact of GPU space constraints we opted for a month Colab Pro subscription that gave us access to the NVIDIA V100 GPU, far more effective than the T4.

7 Conclusion

Our main findings reveal that the full BERT model consistently outperforms the freeze BERT model across multiple evaluation metrics, showcasing the importance of fine-tuning BERT parameters during training. Furthermore, both BERT models significantly outperform baseline classifiers, indicating the efficacy of leveraging pre-trained language representations for sequence classification. While our results align with expectations regarding the superior performance of the full BERT model, some aspects were unexpected. For instance, the freeze BERT model's performance, while lower than that of the full BERT model, still surpassed expectations, indicating that even frozen BERT embeddings can capture meaningful information for sequence classification tasks. Additionally, the consistent outperformance of both BERT models compared to dummy classifiers underscores the robustness and generalization capabilities of BERT-based approaches. However, one major constraint is the reliance on a single pre-trained BERT model (BERT base uncased), limiting the exploration of other BERT variants or alternative pre-trained language models.

8 Links to external resources

Link to Praveengoci Model on Kaggle: `https://www.kaggle.com/code/praveengovi/classify-emotions-in-text-with-bert`
Link to the dataset: `https://lcs2.in/SemEval2024-EDiReF/`.

References

Taewoon Kim and Piek Vossen. 2021. Emoberta: Speaker-aware emotion recognition in conversation with roberta.

Shivani Kumar, Shubham Dudeja, Md Shad Akhtar, and Tanmoy Chakraborty. 2023. Emotion flip reasoning in multiparty conversations.

Shivani Kumar, Anubhav Shrimal, Md Shad Akhtar, and Tanmoy Chakraborty. 2021. Discovering emotion and reasoning its flip in multi-party conversations using masked memory network and transformer.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. Meld: A multimodal multi-party dataset for emotion recognition in conversations.

Xiangyu Qin, Zhiyu Wu, Jinshi Cui, Tingting Zhang, Yanran Li, Jian Luan, Bin Wang, and Li Wang. 2023. Bert-erc: Fine-tuning bert is enough for emotion recognition in conversation.