SemEval 2024 Task 10: Emotion Discovery and Reasoning its Flip in Conversation (EDiReF)
NLP Course Project

Daniel Bernardi, Daniele Santini, Hiari Pizzini Cavagna and **Muhammad Saleem Ghulam**
Master's Degree in Artificial Intelligence, University of Bologna
{ daniel.bernardi, daniele.santini2, hiari.pizzinicavagna, muhammad.ghulam }@studio.unibo.it

Abstract

We challenged the SemEval task about Emotion Recognition (ERC) and Emotion Flip Reasoning (EFR) in English Conversations. For this task, a system should be able to classify the emotion of each sentence and if that sentence caused an emotion flip in the conversation. We developed a PyTorch model based on BERT for both emotion and trigger prediction. The proposed solution demonstrates promising performance in the ERC task. However, its performance on the EFR task is not good enough. To make an attempt in overcoming the emotion flips detection problems, we tried to balance the training set by using a WeightedSampler, or to increase the weight of the trigger prediction in the loss function. Another tentative for balancing the dataset was based on the Easy Data Augmentation approach, by using the *nlpaug* library. These solutions didn't improve the overall performance of the model as expected. An input encoding with more context, however, gave a small performance improvement.

1 Introduction

Here is a brief description of the two tasks that need to be tackled.

The Emotion Recognition in Conversation (ERC) task is defined as follows: given a dialogue, ERC aims to assign an emotion to each utterance from a predefined set of possible emotions. Seven emotions were defined for this task: *neutral*, *joy*, *sadness*, *surprise*, *disgust*, *anger* and *fear*.

The Emotion Flip Reasoning task is defined as follows: given a dialogue, EFR aims to identify the trigger utterance(s) for an emotion-flip in a multi-party conversation dialogue. The task of recognizing an emotion-flip is not trivial. Let's suppose that two people S_1 and S_2 are involved in the conversation and that we're analyzing an utterance pronounced by S_2. To grasp the eventful presence of an emotion-flip there is need to comprehend and link the emotions of the previous and future sentences.

The importance of these tasks stands in the ability of a Digital System to understand sentences' emotions, and how sentences can drive the course of the dialogue. Digital Systems are becoming more powerful and useful day after day, and are now used for both professional and personal tasks, especially after the recent advances on LLMs. An example of how such a system could be useful can be taken by the videogame *Detroit: Become Human*. The scope of the game is for the player to take actions in a futuristic but realistic simulation of our World. The main characters are Androids, deeply integrated the society because of their humanoid shape and thinking. The Androids usually interact with humans, which could be violent, alterated or sad, and the conversation development may change drastically based on the Android responses. Even if we're stretching the reality and entering the fictional world, for those Androids, a much more evolved ERC and EFR system could be fundamental to understand emotions and the dialogue's development. Another possible use case would be for Police Forces, which could make use of such a system to verbally interact in the safest way during problematic situations.

Emotion Recognition is a task similar to Sentiment Analysis, which is not new in the Natural Language Processing world. Many systems were developed to solve these tasks. Usually, the crucial point is to understand emotions based on word usage and meaning. If the solution requires to understand deeply a sentence, Language Models (e.g. Transformers based LMs) may provide a powerful, generic and good solution.

On the contrary, an EFR system cannot base its answer on a single sentence. In fact, being able to understand if the emotion changes in a dialogue requires a system that perceives the evolution of the emotions, sentence-by-sentence. The previous and future context of a sentence gains crucial importance. A transformer based solution seems required.

Our approach to solve the task is to fine-tune a BERT model (Devlin et al., 2019) with two classification heads, which give the model the ability to output both an emotion and the trigger value for each sentence of a dialogue, as by project's guidelines. More specifically, we use a pre-trained BERT checkpoint (bert-base-uncased) as the main module of our model. After some experiments and online research, we decided to concatenate the last 4 hidden layers' outputs as input for our linear classification layers. Since the BERT checkpoint was trained for the Next-Sentence Prediction task, we decided to create an input for each dialogue's sentence, by defining the current sentence, the next, and the previous sentence(s) (the history). Our inputs for BERT are a concatenation of these texts, which provide the necessary context for the two tasks.

The two BERT setups, freezed base-model and Fully finetuned model, were tested on four different configurations. The goal was to try improving the performance of the baseline, more specifically for the EFR task. One of those configuration was standard, while the others included the use of a Weighted Sampler to balance the dataset (trigger-wise), or a loss calculation that enhanced the importance of correct trigger predictions, or the use of the *nlpaug* library to augment the training set, increasing the support of emotion flips.

At the end of these experiments we concluded that these solution could not result in a model with better overall performances. Instead, we obtained the best improvements by giving more textual context to the BERT model.

2 Background

2.1 Weighted Sampler

The WeightedSampler is a PyTorch class. More specifically, it is a sampler utilized by a DataLoader to extract inputs from a dataset. It is possible to specify, for each input row, its probability to be extracted. This provides a solution to the imbalanced nature of some datasets. For the EFR task we tried this approach by enhancing the emotion flipping sentences' probability to get extracted.

2.2 Easy Data Augmentation

Easy Data Augmentation (Wei and Zou, 2019) is an approach for the augmentation of textual datasets. The augmentation is based on the elaboration of sentences at word-level, with operations of random *synonym replacement*, *insertion*, *removal* and *swap*.

2.3 nlpaug

The *nlpaug* library (Ma, 2019) implements many classes for data augmentation of textual and audio sources. It is possible to do operations on Characters, Words and Sentences.

Contextual Word Embeddings Augmenter Augmenter that leverage contextual word embeddings to find top n similar word for augmentation. It can be used to implement the Easy Data Augmentation's approach while making use of a model such as BERT.

3 System description

The loading and the processing of the dataset is done by using the Pandas library. A Label Encoder from scikit-learn is used to convert the emotion labels from categorical data into numerical representations. We have implemented and trained the model using the PyTorch library. The HuggingFace Transformers library provides the necessary tools to encode the data, and also to load and train the pre-trained BERT model. A pre-trained Tokenizer and a pre-trained Model are loaded from the library using the *bert-base-uncased* checkpoint.

We utilized the dataset provided by SemEval, which contains different conversations, and each of them is labeled with a list of emotions, each for each utterance, and a list of binary triggers, that indicates if the utterance triggers an emotion flip inside the dialogue. After the loading, the initial step is the pre-processing of the data. For processing, encoding and training reasons we decided to decompose the dialogues into single utterance, and assigning them to their corresponding labels for emotion and trigger. Furthermore for each utterance we added two additional features, the history and the future, containing respectively previous two sentences and one next sentence of the same dialogue. These additional features are required for a better understanding of the context during training by the model, in a way to improve the overall performance. After the pre-processing and splitting, the input data is encoded using the Tokenizer, by concatenating history, utterance and future features, and then each input sequence is converted into a sequence of tokens *([CLS] history+utterance [SEP] future-utterance)*. The PyTorch Dataset and Dataloader are used to prepare the encoded data for the training.

We implemented a transformer-based model, specifically the pre-trained BERT (Bidirectional Encoder Representations from Transformers). A fine-tuning process is implemented to adapt the BERT model to our specific tasks, by adding a classification head on top of the BERT architecture. As we want to predict two features, we are performing a multi-task-learning, by sharing the pre-trained BERT base layers and a Linear layer, and then feeding separate classification heads for each task. In our case the output of BERT architecture is passed through a Linear layer, and then we have the following two streams:

- For the prediction of emotions, we have a Tanh function, a Dropout layer and a Linear layer followed by a Softmax function (in PyTorch is incorporated in the CrossEntropyLoss). As just mentioned the criterion for computing the loss is the CrossEntropyLoss ;
- For the prediction of triggers, we have a ReLU Function, a Dropout layer and a Sigmoid function on a single output of a Linear layer. In this case the loss is computed by using the BinaryCrossEntropy criterion.

An implementation of the network heads is shown in Figure 1.

BERT produces different outputs that can be used for classification tasks, such as the pooler output, the sequence output and the hidden states. The pooler output contains a summary representation of the sequence, and it's basically the [CLS] token of the last hidden state passed through a linear layer with a non linear activation function. The sequence output contains an embedding for each token in the input sequence. Instead the hidden states refer to the outputs of each layer of BERT, by making possible to aggregate information from multiple layers. There are different configurations that can be used depending on the tasks, after some experimentation we decided to take the mean between the [CLS] tokens of the last four hidden states, and feed the top layers. In our case this configuration turns out to perform slightly better than other alternatives, such as the pooler outptut or the concatenation of the last four hidden states.

4 Data

The data used in this project was given by SemEval and is freely accessible online, the link is provided in Section 8. The data is formatted as a list of dictionaries, each of them containing 5 keys: *episode, speakers, emotions, utterances, triggers*. The utterances value contains the dialogue between the speakers in a list format, splitted at the sentence level. The emotions and the triggers values are the labels used for the classification tasks, respectively string labels and float labels (0.0 or 1.0).

The data is loaded into a Pandas DataFrame and splitted in training, validation and test sets. The pre-processing steps we followed are the following:

- Substitution of wrongly-loaded trigger values, from *None* to *0.0*.
- trigger label formatted as boolean values.
- dropped the *speakers* column.
- after performing the data split into training, validation and test set, we splitted each *episode*, creating a row in the DataFrames for each utterance of a dialogue.
- created the *history* and the *future* columns. The former included a concatenation of two previous sentences of the utterance, the latter contained the next sentence.

Some analysis on the distribution of data showed that the distribution of emotion classes is imbalanced (see Figure 2). Furthermore, 29425 sentences out of 35000 ($\sim$84.01%) didn't cause an emotion flip in the conversation, meaning that the presence of positive triggers is quite rare.

One of the implemented systems performed an additional pre-processing step of the data, with the goal of improving the trigger prediction's F1 score. We extended the training set by using the **ContextualWordEmbsAug** class provided by the *nlpaug* library. We created a pipeline of textual operations to be performed: insertion, swapping and random deletion of words. We also needed to

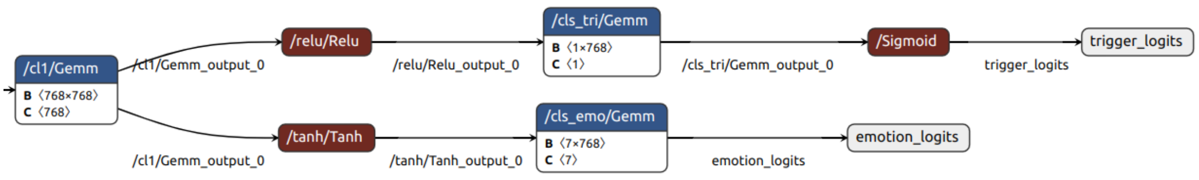


Figure 1: Classification heads implementation on the top of BERT.

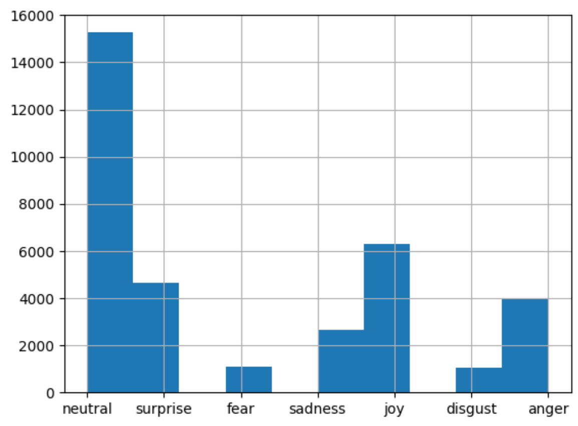


Figure 2: Emotions distribution in the dataset

specify the probability of the operation happening, and the Contextual Word Embedding that would be used by the class (bert-base-uncased). The insert and swap operators made use of context around words to add or swap them based on similarity.

Since we decided to encode the history, the current utterance, and the next one in the input for the BERT model, each one of these strings were augmented with the operations explained before.

The inputs of the augmentation pipeline were all the DataFrame's rows showing an emotion flip. For each of these training set sentences, we created and added two other similar variants, and tripled the support of emotion flip sentences in the training set. The validation and test splits remained untouched. An example of augmented utterance can be seen in Table 1.

Table 1: Sentence augmentation example

	sentence
original	Hi, listen, I'm sorry about before.
aug 1	hi, um listen, i'm sorry tonight.
aug 2	so hi, listen, i'm sorry earlier.

5 Experimental setup and results

The execution and analysis of the model was performed using a python notebook which allowed to customize the hyper-parameters of the execution:

- Learning rate: After some experimentation we settled to $5 * 10^{-5}$ as higher values cause worse performances
- Seed: a number to use as seed for pseudo-random number generation, pyTorch and CUDA. The five numbers chosen to be used as seeds were 11, 27, 35, 42 and 81.
- Weighted sampling: it is possible to enable or disable the usage of a weighted sampler in the training data loader (as described in section 2.1)
- Trigger weight: number used as trigger loss scaling factor. We choose values equal or lower to 3 as higher values cause worse performance.
- Data augmentation: whether to perform data augmentation with nlpaug (as described in section 2.3 and 4)

The notebook includes all the necessary steps for executing and evaluating the model:

- Dataset ingestion, pre-processing, splitting and (if enabled) augmentation (as described in 3)
- Testing of the baseline model
- Training and testing of the model with the freezed base-model
- Training and testing of the model with the fully-finetuned model
- Programmatic error analysis on the output of the model
- Dump of the models' output and F1 scores to files for further manual analysis

For each model configuration tested we used as metrics the sequence F1 score (average of the dialogues' scores) and the unrolled sequence F1 score (average of the flattened utterances scores), for both emotions and triggers.

In order to prevent overtraining the models were trained with early stopping, halting training if the validation loss of an epoch is higher than the last.

Four combinations of hyper-parameters were chosen to test the model behavior:

- S1: normal sampling, trigger weight = 3.0, no data augmentation
- S2: weighted sampling, trigger weight = 1.0, no data augmentation
- S3: normal sampling, trigger weight = 1.0, no data augmentation
- S4: normal sampling, trigger weight = 1.0, data augmentation enabled

The notebook was run for each combination of hyper-parameters and seed, with the results reported in Appendix A. A recap of F1 scores for triggers can be found in Table 2.

6 Discussion

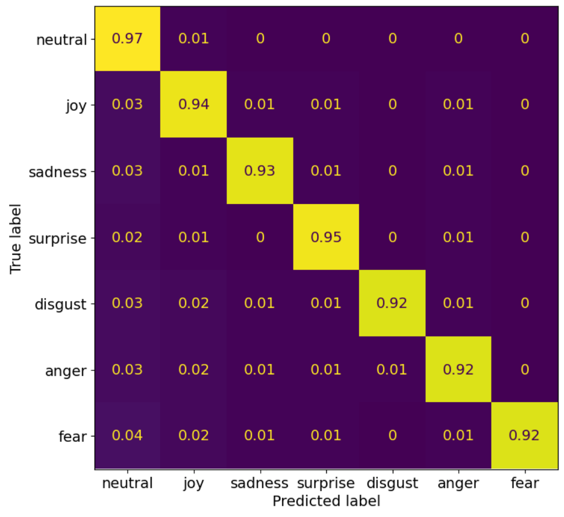
6.1 Baselines

Two baseline classifiers, namely the Random Classifier and the Majority Classifier, were employed in this study. The Random Classifier, owing to its inherent nature, serves as a weak benchmark for emotion classification, yielding an SEQ-F1 micro score of 0.150 for emotions and 0.510 for triggers. On the other hand, the Majority Classifier, which outputs the most frequent label, provides a more substantial baseline due to the imbalanced dataset. It achieves an SEQ-F1 micro score of 0.420 for emotions and a notably higher 0.809 for triggers.

The significance of the Majority Classifier as a baseline is particularly important as a reference, given the highly unbalanced nature of the dataset, in particular for the binary triggers classification. In binary classification, achieving a commendable F1 score can be relatively straightforward by consistently predicting the most frequent label (0).

6.2 BERT models results

The Freezed and Full BERT models, configured with the initial settings (S3), demonstrated good performance in emotion classification and an equal



	neutral	joy	sadness	surprise	disgust	anger	fear
neutral	0.97	0.01	0	0	0	0	0
joy	0.03	0.94	0.01	0.01	0	0.01	0
sadness	0.03	0.01	0.93	0.01	0	0.01	0
surprise	0.02	0.01	0	0.95	0	0.01	0
disgust	0.03	0.02	0.01	0.01	0.92	0.01	0
anger	0.03	0.02	0.01	0.01	0.01	0.92	0
fear	0.04	0.02	0.01	0.01	0	0.01	0.92

Figure 3: Emotion Classification, normalized confusion matrix - Freezed BERT, S3 settings

performance in trigger classification, compared with the Majority Classifier. Moreover, we didn't notice a significant difference between the results of the Freezed and Full BERT models, with a slight advantage in favor of the Freezed BERT model. Consequently, in the following discussion it will be used as our trained model reference.

6.2.1 Emotion Classification

Both the Freezed and Full BERT models achieved really similar SEQ-F1 micro scores of 0.944 and 0.946, respectively. These scores represent a substantial improvement over the baseline models. As shown in Figure 3 the neutral label is the most accurately predicted, while anger, fear, sadness and disgust show a slightly higher miss-classification rate. This could be attributed to the imbalance of the dataset, without a particular pattern to be notice. In Figure 4 is possible to observe a correlation between the label support and the F1 score, which may explain the differences in scores, rather than an intrinsic label complexity. In fact, the neutral emotion, having the best support, exhibits the best F1 score, whereas disgust, anger and fear the lowest scores.

6.2.2 Trigger Classification

On the contrary, when it comes to trigger classification, the models did not exhibit a significant enhancement compared to the Majority Classifier baseline, recording SEQ-F1 micro scores of 0.806 and 0.807. As shown in Figure 5 the Freezed BERT model demonstrated an high accuracy in the classification of sentences without a trigger. However,

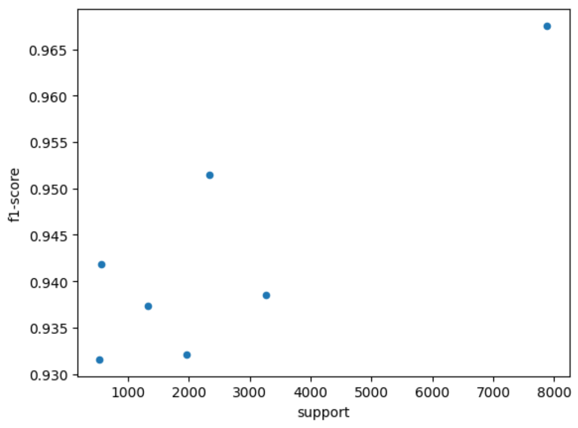


Figure 4: F1 score and Support, Emotion Classification - Freezed BERT, S3 settings

its capability in identifying triggers was notably low, with a tendency to miss-classify the presence of triggers in the majority of cases.

These similar performances between the trained models and the Majority Classifier may be attributed to the challenges posed by the highly imbalanced dataset and the inherent difficulty in identifying triggers within sentences.

6.2.3 Trigger Classification improvements

Given the discussed results, we attempted various approaches in order to improve the trigger classification. Table 2 provides a comparison of the F1 scores for these attempts, while more detailed tables are presented in the Appendix.

S1 settings: Our initial attempt involved assigning greater importance to the trigger loss during training. Both trained models exhibited a slight improvement in the F1 score for True triggers while maintaining the same score for False triggers classification. However, a minor decrease in the F1 score for emotion classification was observed. In summary, the overall triggers UNR-F1 and SEQ-F1 micro scores remained constant, with a slight improvement in the UNR-F1 macro scores.

S2 settings: Moreover, we experimented the use of WeighterSampler to boost the trigger sentences' probability to be extracted. This approach led to a significant difference in the triggers F1 scores. Specifically, the False score declined from 0.913 to 0.776, but the True score had an important improvement, soaring from 0.132 to 0.296. Consequently, both the UNR-F1 and SEQ-F1 micro score triggers score declined, but the UNR-F1 score macro slightly improved.

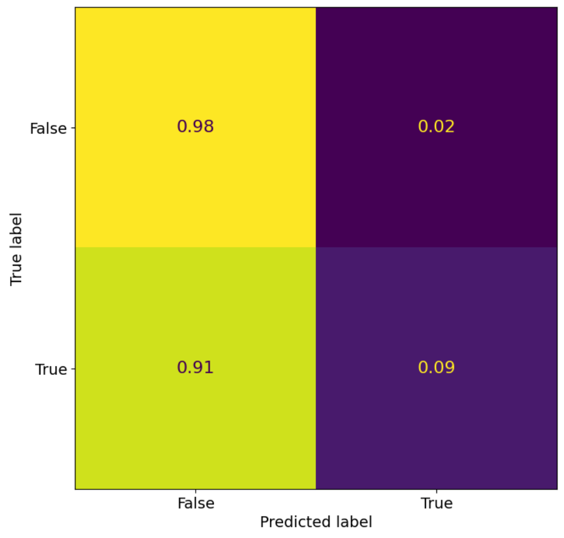


Figure 5: Trigger Classification, normalized confusion matrix - Freezed BERT, S3 settings

S4 settings: Finally, we tried a data augmentation approach, with the purpose of increasing the sentences containing triggers in the dataset. However, this approach didn't led to any notably improvement in the outcome, resulting in a slightly worsening variation of the F1 score of the True trigger label.

6.2.4 Encoding without the future sentence

Initially we performed some tests by using a different encoding, that concatenated the *history* and the current *utterance*, without considering the *future* sentence. That encoding was then changed, because including the future sentence improved the EFR task's Unrolled F1 Score macro by 2 points.

Table 2: Triggers F1 scores

	Freezed BERT		Full BERT	
Trigger	False	True	False	True
S1	0.913	0.150	0.913	0.171
S2	0.776	0.296	0.789	0.289
S3	0.913	0.132	0.912	0.148
S4	0.913	0.108	0.912	0.104

7 Conclusion

This project explored and showed the performance of a BERT-based model for the ERC and EFR tasks, comparing it with a Random and a Majority Classifier baseline. Our model, trained on S3 settings, significantly improved the ERC task, compared to the two baselines. However, even if the SEQ-F1 and UNR-F1 for the emotion flip prediction improved significantly, it cannot be said to be good enough.

Some tests based on different representations of the input were made to decide which sentences could provide a good enough context for both tasks, that resulted in the previously explained encoding (*history+utterance [SEP] future-utterance*).

Our work then focused in finding alternative solutions for the model's training, with the goal of improving its emotion flip detection capabilities. The S1 and S2 solutions provided better EFR performance, at the cost of decreasing the emotion prediction's score. The S4 solution, using data augmentation to extend the dataset, proved to be not useful for this goal.

In conclusion, neither of the previous solutions let untouched the ERC performance while improving the EFR score. The most important findings that we obtained is about the input encoding. In fact, an encoding without the next sentence, resulted in having more or less the same scores in the ERC task, but worse performance for the EFR task.

Since the best improvement was obtained by changing the encoding, exploring more that subject could be useful for searching better solutions. Additional enhancements may involve experimenting with an expanded version of BERT or implementing distinguished BERT variants, such as RoBERTa or XLNet.

8 Links to external resources

- Dataset download (Task 3 folder)
- Github Repository with code. You should follow the *execution.ipynb* Notebook.
- Netron App was used to create an image of our model's classification heads.
- Google Colab and Kaggle Notebooks were used to run the code.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

A Appendix

Baseline	Random		Majority	
	Average	σ	Average	σ
Sequence F1 emotions micro	0.150	0.004	0.430	0.007
Sequence F1 triggers micro	0.510	0.008	0.809	0.012
Unrolled Sequence F1 emotions micro	0.147	0.004	0.441	0.010
Unrolled Sequence F1 triggers micro	0.507	0.006	0.847	0.010
Unrolled Sequence F1 emotions macro	0.123	0.003	0.087	0.001
Unrolled Sequence F1 triggers macro	0.437	0.008	0.458	0.003

S1: trigger weight = 3.0	BERT Full		BERT Freezed	
	Average	σ	Average	σ
Sequence F1 emotions micro	0.927	0.022	0.931	0.001
Sequence F1 triggers micro	0.809	0.016	0.808	0.010
Unrolled Sequence F1 emotions micro	0.936	0.018	0.941	0.003
Unrolled Sequence F1 triggers micro	0.842	0.013	0.843	0.009
Unrolled Sequence F1 emotions macro	0.917	0.026	0.924	0.003
Unrolled Sequence F1 triggers macro	0.541	0.015	0.531	0.015

S2: weighted sampling	BERT Full		BERT Freezed	
	Average	σ	Average	σ
Sequence F1 emotions micro	0.916	0.017	0.921	0.007
Sequence F1 triggers micro	0.667	0.018	0.656	0.008
Unrolled Sequence F1 emotions micro	0.926	0.015	0.932	0.005
Unrolled Sequence F1 triggers micro	0.674	0.016	0.660	0.009
Unrolled Sequence F1 emotions macro	0.907	0.020	0.916	0.004
Unrolled Sequence F1 triggers macro	0.538	0.008	0.536	0.008

S3: standard	BERT Full		BERT Freezed	
	Average	σ	Average	σ
Sequence F1 emotions micro	0.944	0.006	0.946	0.007
Sequence F1 triggers micro	0.806	0.011	0.807	0.011
Unrolled Sequence F1 emotions micro	0.952	0.004	0.953	0.005
Unrolled Sequence F1 triggers micro	0.841	0.008	0.842	0.009
Unrolled Sequence F1 emotions macro	0.943	0.008	0.942	0.008
Unrolled Sequence F1 triggers macro	0.529	0.019	0.522	0.022

S4: with data augmentation	BERT Full		BERT Freezed	
	Average	σ	Average	σ
Sequence F1 emotions micro	0.935	0.009	0.942	0.004
Sequence F1 triggers micro	0.803	0.011	0.806	0.014
Unrolled Sequence F1 emotions micro	0.944	0.008	0.951	0.002
Unrolled Sequence F1 triggers micro	0.839	0.010	0.842	0.011
Unrolled Sequence F1 emotions macro	0.932	0.010	0.940	0.005
Unrolled Sequence F1 triggers macro	0.507	0.018	0.511	0.018