Emotions and Triggers recognition in multiparty conversations deploying BERT-based architectures.

Analysis of how BERT-based models perform in Emotions and Triggers recognition tasks, comparing performances obtained by fine-tuning only the classification heads and the whole architecture.

NLP Project

Salvatore Guarrera, Riccardo Monaco
Master's Degree in Artificial Intelligence, University of Bologna
{salvatore.guarrera, riccardo.monaco2}@studio.unibo.it

Abstract

The work conducted represents a case-study of how to classify utterances of several dialogues, based on the expressed emotions and the triggers, meaning the change of emotion in the interlocutor. To do so, we used a popular architecture in the NLP world, the BERT model, which helped us obtain satisfying results. The main takeaway has been the huge difference that fine-tuning the whole architecture can make with respect to freezing the backbone weights and fine-tuning only the classification heads on top of it.

During our project completion, we came across a few challenges, especially finding the right way to encode the input to the BERT model, and finding the best hyperparameters possible for training.

1 Introduction

The project we chose to pursue takes on the challenge "Emotion Discovery and Reasoning its Flip in Conversation (EDiReF), SemEval 2024 Task 10", seeking to classify speakers' emotions and triggers in multiparty conversations. The possible approaches to tackle this problem were various, LSTMs and Transformers among others. The required solution for the project was BERT, using bert-base-uncased as base model, a variation of BERT where there are no differences between cased and uncased words (eg. NLP is the same as nlp), with 2 classification heads for each classification task (*emotions* and *triggers*).

After a quick analysis and pre-processing of the dataset, we encoded input and output in the best way to work with BERT, especially the input, considering the constraint of 512 as maximum length. We, then, built a model based on BERT, attaching 2 classification heads, one for each classification task, and we actually trained and evaluated two different settings of the model, one where the BERT embedding layer weights were frozen, fine-tuning only the classification heads, and one where the fine-tuning was done on all the model. In order to perform a hyperparameter tuning, we used the KerasTuner random search approach in order to get baseline values from which to start for a manual tuning. The KerasTuner was run only on the model with the frozen BERT and the same hyperparameters values were used also for the training of the unfrozen BERT, in order to do a more coherent comparison of the results of the two models. Finally, we compared the two BERT, and we compared them also with the two baseline models, a random and a majority classifier, using two metrics for emotions and triggers labels: Sequence F1, reporting the average of the f1-scores computed for each dialogue, and Unrolled Sequence F1, computing the f1-score once having flattened all the utterances.
We trained the model on five different seeds, showing how robust the model we built is and how sound the evaluation is. The results obtained were extremely satisfying, highlighting the much higher performances of the model fine-tuning also BERT's weights (the *unfrozen* one), reaching an Unrolled F1 score of 0.945899 in the Emotions recognition and an Unrolled F1 score of 0.649405 in the Triggers recognition, as average of the results obtained among the 5 seeds.

These results showed us how important it is to have an already high performing backbone, but also how important it is to train the backbone together with the whole architecture in order to obtain really satisfying results, which can be further enhanced with a few improvements.

2 System description

After some pre-processing done on the dataset provided, we went through with the tokenization of the *utterances* using the AutoTokenizer of the *transformers* library. Like we said before, the main character of this project is the BERT model, so, for

our experimental setup, we created a customized BERT-based architecture, with two classification heads on top of BERT, one for triggers and one for emotions recognition.

Let's dive into the model. First, it's worth noting that we decided to proceed deploying TensorFlow, for our own preference. Unfortunately, at this point in time (Nov '24), a few incompatibilities between the last versions of the libraries *transformers* and *tensorflow* arose. After many trials, we were able to find compatible versions: the 4.37.2 and the 2.15.1, respectively. The backbone of our model is the bert-base-uncased, where BERT is initially frozen, with its embedding layer weights fixed during training, in order to focus on adapting higher layers and fine-tune the classifier heads, and, then, unfrozen, fine-tuning the whole model architecture.

We created the input_ids and the attention_mask, which, once flattened for compatibility with BERT's input requirements, will represent the input to our model. As the output of the BERT model, we extracted the last_hidden_state, which is the output tensor encapsulating the contextual embeddings for each token, in a 768-dimensional space, with a shape of (batch_size, max_dialogue_length * max_utterance_length, 768).

In order to reduce dimensionality and capture dialogue-level representations, we apply a 1D max-pooling operation across the max_dialogue_length dimension, yielding an output shape of (batch_size, max_dialogue_length, 768) which aligns with our labels. This pooling approach to the last_hidden_state is optimal for our task, as it avoids BERT's next-layer output, which is intended for whole-sentence embeddings, incompatible with the purpose of this project.

What comes as output of this MaxPooling1D layer goes as input to the two classification heads. Each of those follows the same structural pattern, starting with a Dense layer with ReLU activation, followed by a dropout layer. Additional Dense and Dropout layers with ReLU activations may be appended, contingent on hyperparameter optimization. The hyperparameters tuning is done using the KerasTuner random search, where we set as hyperparameters to be found the dimensionality of the Dense layers and the number of hidden layers, for each of the two heads. The values of the hyperparameters tuning and the various experiments done will be analyzed in Section 4.

For each classification head's final output layer, we apply a softmax activation with the dimension set to triggers_len for trigger classification and the respective emotion classes for emotion classification.

The model has obviously two outputs, representing predictions for triggers and emotions, respectively.

The model architecture is shown in Figure 5.

Finally, our model was compiled with an Adam optimizer, with the learning rate to be finalized in the hyperparameter tuning. For the loss function, we used categorical cross-entropy for the two heads, as they are both one-hot encoded. Metrics of interest include the accuracy, in order to evaluate classification performance across both tasks.

3 Data

As already mentioned in the introduction section, we took the dataset from the "Emotion Discovery and Reasoning its Flip in Conversation (EDiReF), SemEval 2024 Task 10" challenge. We performed pre-processing in order to have the data in the format most suitable to complete our task. The initial five columns of the dataset were 'episode', 'speakers', 'emotions', 'utterances', 'triggers', but we put the most emphasis on the last three, the only ones we actually needed.

3.1 Utterances column

First of all, it was important to check for any missing values (Nan or None), but we didn't find any. So, we could go ahead and tokenize the dialogues with the BERT tokenizer provided by the transformers library, AutoTokenizer, and plot histograms to visualize the frequencies of the dialogues lengths and the utterances lengths (Figures 1 and 2).

We noted that the distribution was skewed towards smaller values, gradually decreasing as the lengths increased. Thanks to this analysis, we decided to compute the percentiles (0.25, 0.50, 0.75, 0.90, 0.95, 1.00) in order to have a clearer selection of the lengths we wanted to use for the encoding.

3.2 Emotions column

We moved forward checking for missing values in the emotions column, not finding any. Though, we found out that the distribution was also skewed. In fact, the 'neutral' emotion came up twice as many times as the second most frequent emotion, 'joy'. Figure 3 shows this behavior.

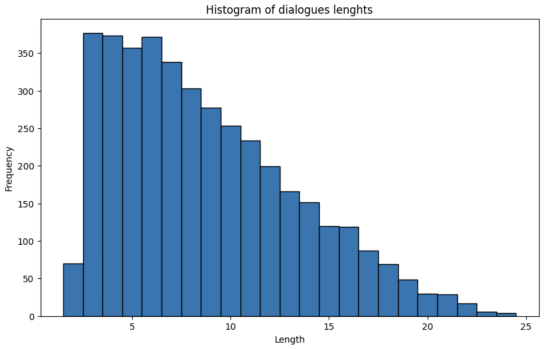

Figure 1: Distribution of frequencies of the lengths of the *dialogues*

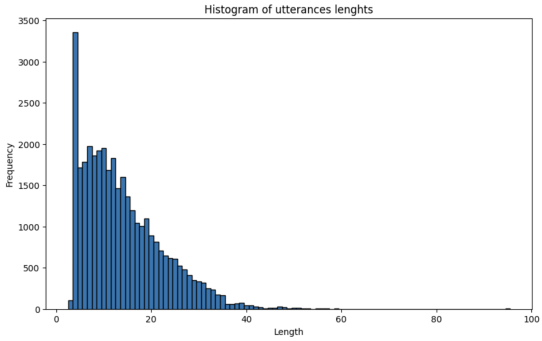

Figure 2: Distribution of frequencies of the lengths of the *utterances*

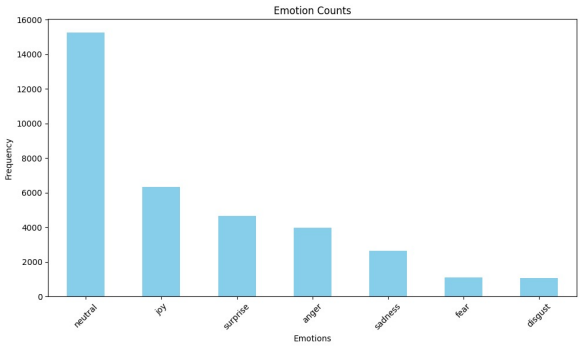

Figure 3: Distribution of frequencies of *emotions*

3.3 Triggers column

Finally, we analyzed the triggers column, finding some None values, and changing them to 0's. The histogram of the frequencies of the triggers in Figure 4 highlights a deep unbalance between the positive trigger (1.0, where there is an actual *emotion* flip) and the negative trigger (0.0, where no *emotion* flips happen). This deep unbalance is the main reason for poor performances, as shown later in the results.

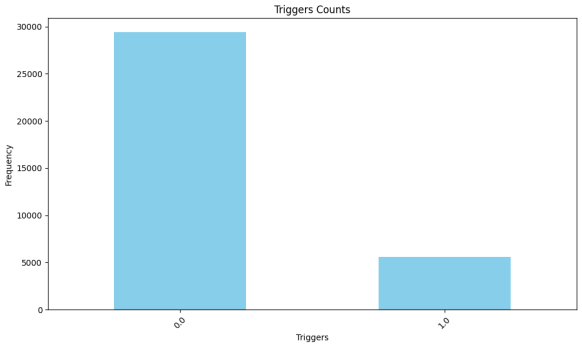

Figure 4: Distribution of frequencies of *triggers*

3.4 Data split

According to the 80/10/10 proportions suggested in the project guidelines, we split the data into train, val and test sets, ordering the indexes of the three sets, for a more coherent visualization of the data.

3.5 Encoding

3.5.1 Input

Given that the maximum input tokens of the BERT is 512, we chose 16 as maximum dialogues length and 32 as maximum utterances length, resulting in exactly 512 tokens (16*32 = 512) once flattened. The two values represent, respectively, the 90th and about the 95th percentile of the two lengths.

The encoding of the model input is done through the tokenization, deploying BERT's tokenizer, of the utterances, which outputs two arrays for each set (train, val, test), filled with *input_ids* and *attention_mask*. Then, we proceeded to the padding or truncation of both the tokenized dialogues and utterances with a length bigger or smaller, respectively, than the maximum lengths decided earlier. The same thing is done simultaneously for the attention mask.

Finally, the encoded input has size (#EXAMPLES, MAX_DLG_LEN, MAX_UTT_LEN)

3.5.2 Output

The encoding of the output of the model is done simply by one-hot encoding emotions and triggers.

- For the emotions, we first need to create a dictionary and add the token <PAD>, to represent the utterances only made of 'padding'.

- For the triggers, the padding element is represented by the value '2'.

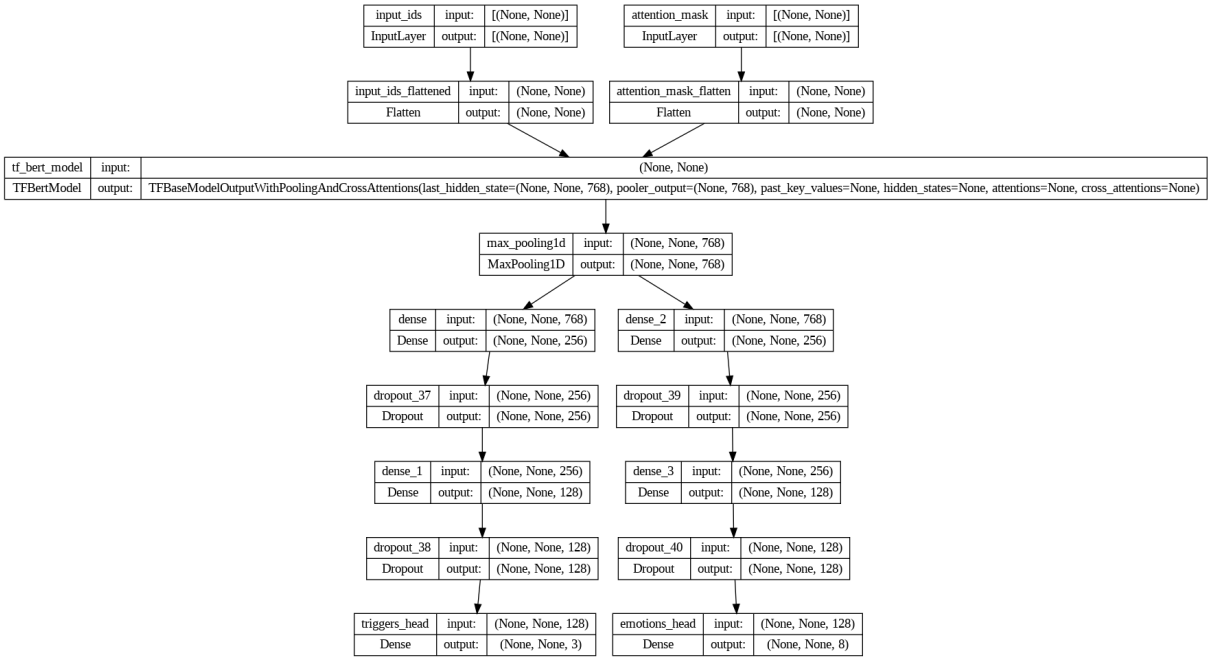

Figure 5: BERT-based model architecture

Emotions and triggers can, then, be one-hot encoded into numpy arrays, also taking care of the eventual padding or truncation.

4 Experimental setup and results

As shown in Figure 5, the architecture is a straightforward BERT-based model for a multi-head classification task. In order to find the best values for the number of activations of each layer, number of hidden layers of each head, and the learning rate, we made use of the *KerasTuner* for the first stage of the hyperparameters tuning, using those values as the starting point.

The KerasTuner lets select between several search algorithms, among which we chose the RandomSearch, with the *validation loss* as the objective to be optimized. After five trials, the best hyperparameters found were:

- Head1 layer1 activations: 55
- Head1 num layers: 0
- Head2 layer1 activations: 110
- Head2 num layers: 0
- Learning Rate: 0.001

We also chose to keep the same values obtained from the hyperparameters tuning of the *Frozen* BERT model also for the *Unfrozen* BERT model, in order to perform a more coherent analysis of the results obtained.

Using these values, we noticed that the results were not satisfying at all. Since we only used the KerasTuner for the baseline values, we could iterate and carry on several trials in order to get better results. We "played" especially with the number of hidden layers and the number of activations of each layer. After a few disappointing trials, where both the classification heads suffered from underfitting, we brought down the Dropout rate, initially set to 0.5, to 0.1, making the model a bit more complex, adding an hidden layer to both the heads. Results kept improving until we found the configuration that worked best and that gave really impressive results:

- Head1 layer1 activations: 256
- Head1 num layers: 1
- Head1 layer2 activations: 128
- Head2 layer1 activations: 256
- Head1 num layers: 1
- Head1 layer2 activations: 128
- Learning Rate: 0.0001

The training was carried out using the Adam optimizer, with the learning rate determined as above,

and the loss for both the classification heads was the categorical cross-entropy. The choice of this specific loss was taken due to the fact that the classification heads' outputs are one-hot encodings. Finally, about the metrics, the chosen one was the accuracy.

The results obtained with this configuration in the *frozen* model were promising, showing a much higher F1, both Sequence and Unrolled, than the one obtained from the Random Uniform and Most Frequent classifiers used as the baseline models. Though, the really impressive results were obtained by deploying this configuration with the *unforzen* BERT model. Table 1 shows our findings.

5 Discussion

For an exhaustive discussion of the work conducted, we analyzed quantitative results of the models, based on the F1 metrics taken into account, and what we could have done differently and could be improved with further works.

The first and most basic analysis that can be carried out is the one which involves the behavior of the BERT-based models with respect to the baseline models. All the former models overcome the latter. This is comforting, meaning that the model we built on top of BERT performs better than Random Uniform and Most Frequent classifiers. Speaking of the two BERT-based models, we see a deep difference between their behavior. In fact, the *unfrozen* one performs much better than the *frozen* one, both for *emotions* and for *triggers*. The gap is enormous, with the Unfrozen BERT obtaining an Unrolled F1 score almost twice as higher than the Frozen one does. Same thing happens for the *triggers* classification head, where the gap is smaller, but still noteworthy. Together with the F1 scores, we computed their standard deviations. The values show the robustness of our results, where the worst case, the triggers head using Unfrozen BERT, still is one degrees of magnitude lower than the average F1 between the five seeds.

This big distance between the F1's obtained from the two BERT-based models highlights how much an influence BERT has in the training. For the sake of comparison, we plotted the confusion matrices of the Emotions classification using the two models. Figure 7 clearly shows how well the *Unfrozen* model works.

As follows, a discussion about errors made by the models and what we believe caused these unwanted results. First of all, since the results over the five different seeds were pretty similar, we only focused our error analysis on the results of one single seed. We chose to carry on the analysis deploying classification reports showing *precision*, *recall* and *F1 scores*, class-wise, and confusion matrices. Let's split the analysis following the two classification heads.

5.1 Emotions

In the BERT-based Frozen model, the worst classes, meaning the ones which are the least predicted, are *fear* and *disgust*, with 0.00 in all the metrics shown in the Classification report and really low support, 109 and 101, respectively, compared to the total 3451 examples of the test set. The third worst performing class is sadness, also the third least represented class with support of 252, showing the direct correlation between the low frequency of the classes and the low F1-score. This is due to class imbalance, especially with respect to the *neutral* class, which has a support of 1514, more than 40% of total examples.

Taking a look at the Confusion matrix shown in Fig. 6, the *neutral* class is the most predicted, as expected, but it is often confused with the class *joy*, as many as 223 times, due to the fact that they are the most represented in the dataset, showing again how the *frozen* model needs some more work to close the gap with the *unfrozen* one. Focusing on the other classes, the most confused ones are *joy*, *anger* and *surprise*. In particular, *joy* is often confused with *surprise* and *anger*, which is plausible given that even a human could possibly make this kind of mistake, existing words with a double meaning, both positive and negative (eg. the word *terrific*). On the other hand, *surprise* is often mistaken both with *joy* and *anger*, emotions with opposed feelings, given that *surprise* is classified as emotion with *mixed* feeling between positive and negative.

In the BERT-based Unfrozen model, almost all classes are perfectly predicted, indicating a really high performing model. The worst performing class is *sadness*, with a F1-score of 0.91, with a really low precision, "only" 0.86. Unlike the previous model, there is no correlation between F1-score and frequency of the class. The most confused class is *joy* with *neutral*, still showing a slight tendency of confusing the two most represented classes in the dataset.

Model	Sequence F1	Unrolled F1	StDev Sequence F1	StDev Unrolled F1
Most Frequent : Emotions	0.184998	0.087124	0.000000	0.000000
Random Uniform : Emotions	0.093394	0.120788	0.004602	0.002207
Frozen BERT : Emotions	0.410178	0.332040	0.005728	0.007386
Unfrozen BERT : Emotions	0.920198	**0.945899**	0.005150	0.005957
Most Frequent : Triggers	0.515704	0.460950	0.000000	0.000000
Random Uniform : Triggers	0.407323	0.430308	0.005435	0.003189
Frozen BERT : Triggers	0.519374	0.477989	0.006285	0.012576
Unfrozen BERT : Triggers	0.627386	**0.649405**	0.027267	0.030506

Table 1: F1-scores for all the models trained

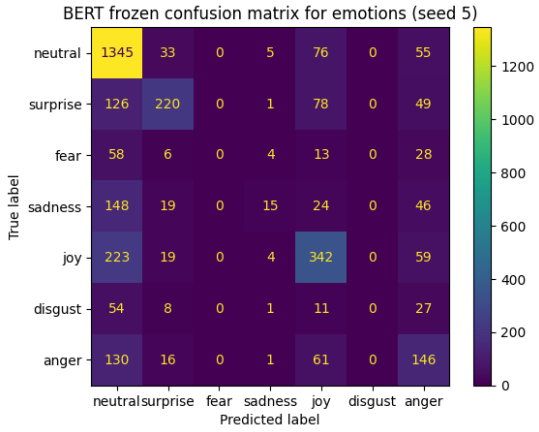

Figure 6: Confusion matrix for the *emotions* recognition using BERT Frozen model

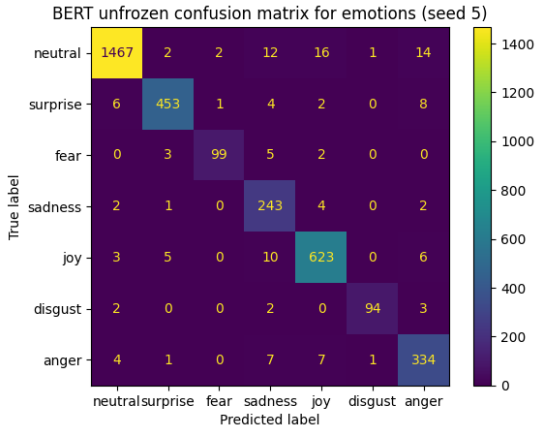

Figure 7: Confusion matrix for the *emotions* recognition using BERT Unfrozen model

5.2 Triggers

Accounting for *triggers* recognition, the results obtained from both the models follow a similar pattern. Performances are not great, but this is given by the fact that the classes are highly unbalanced, 2951 examples of trigger 0.0 versus the 500 examples of trigger 1.0. In particular, the positive trigger's recall in the *frozen* model is extremely low, 0.01, while it increases in the *unfrozen* one (0.39), an improvement also noticeable in the F1-scores, going from 0.03 to 0.42. Confusion matrices also testify the poor performances of the two models, where false negatives are many more than false positives. Though, while the imbalance is pretty obvious in the *frozen* model (493 FN vs 14 FP), it is much less evident in the *unfrozen* one (303 FN vs 243 FP).

Although the *Unfrozen* model is almost perfect when classifying *emotions*, it still needs some work for maximizing *triggers* recognition. Some ideas on how to do so are a deeper hyperparameters tuning on the triggers classification head, which we were not able to do due to a lack of resources, a more balanced dataset where positive triggers are much more represented, and the use of a more performing model backbone than BERT, such as another Transformer model with a bigger input window than the 512 maximum input tokens of BERT.

6 Conclusion

The work conducted showed how big of a difference makes fine-tuning a whole model architecture with respect to fine-tuning only the classification heads on top of a frozen model, hinting that, when possible, it is always better to opt for a *Full* model to obtain high performances. What surprised us is that it didn't take so different time to train an epoch in the two settings (ca. 172s for the *frozen* and ca. 435s for the *unfrozen*), revealing how the second model is the way-to-go. What we thought was our main limitation in obtaining even better performances was the constraint of using the small *bert-base-uncased* model, leaving us curious to compute the same task with a bigger model and seeing if it would reach high performances keeping it frozen.

References

Google BERT model - HuggingFace: https://huggingface.co/google-bert/bert-base-uncased

Kumar, S., Akhtar, M. S., Cambria, E., & Chakraborty, T. (2024). SemEval 2024–Task 10: Emotion Discovery and Reasoning its Flip in Conversation (EDiReF). arXiv preprint arXiv:2402.18944.