Emotion Discovery and Reasoning its Flip in Conversation
NLP Course Project

Gianluca Di Mauro, Leonardo Monti, Luigi Porcelli
Master's Degree in Artificial Intelligence, University of Bologna
{ gianluca.dimauro, leonardo.monti3, luigi.porcelli}@studio.unibo.it

Abstract

Emotion Recognition (ERC) and Emotion Flip Reasoning (EFR) in English Conversations are both NLP tasks of major interest. They consist in classifying the emotion of each sentence in dialogues and understanding if each emotion causes an emotional flip in the dialogue. This report shows how different Bert-based models were able to achieve great results, despite having a not-so-well-organized dataset to work with. The obtained results are presented and compared, analyzing the relationship between performances and different data structure modifications we tried to use to train the models.

1 Introduction

The two tasks analyzed in this report are Emotion Recognition in Conversation and Emotion Flip Reasoning. Both tasks are based on multi-party conversation dialogues. The first task requires assigning an emotion to each sentence of a given dialogue, from a designated set of possible emotions: neutral, joy, sadness, surprise, disgust, anger, and fear. The second task requires understanding whether a sentence in a dialogue triggers a change of emotion. Classifying a trigger requires the understanding of the context of several utterances of dialogue, hence we need more than a simple classification of single items; this topic was tackled with the implementation of sliding context windows.

ERC and EFR are crucial in various real-world scenarios, due to their impact on human-machine interaction. For example, they both cover an important role in virtual assistants, to adapt their responses to user behavior. Definitely worth mentioning the huge impact on the medical field, as ERC facilitates remote emotion recognition and treatment in hospital and home environments (Guo R, 2024). Potential applications of these tasks also span from social media content analysis for mining opinions to the generation of emotion-aware dialogues to make a conversational agent more human-like for several applications, for example in entertainment, like in videogames (Poria et al., 2019b).

Several classic approaches are commonly used to understand the emotions of natural languages, such as ruled-based approaches, key-word-based approaches, and many others, but they fail to fully recognize implicit emotion in the text due to the lack of linguistic information (Alswaidan, 2020). Deep learning techniques, on the other hand, provide the required flexibility adapted to the sequential nature of language by observing the context of the part of the text of interest. The transformer architecture, in particular, was beneficial to handle this kind of task, as it's been repeatedly proven as one of the best options for tasks involving human emotions (Rezapour, 2024). The work presented in this report aims to use a transformer-based approach to perform ERC and EFR. In particular, the focus was on fine-tuning a BERT model (Devlin et al., 2019), completed by two classification heads, giving the model the capability of splitting the output, for emotions and triggers.

2 System description

The system runs entirely on Python. The neural library of choice is PyTorch (Paszke et al., 2019b). First, the data is loaded using Pandas library (pandas development team, 2020), exploring the data to find relevant information, like class imbalances. Then we performed pre-processing and tokenization operations, described in detail in the next paragraph.

The first two models implemented are the baselines, composed of a Random classifier and a Majority classifier, both taking advantage of the DummyClassifier Class (Pedregosa et al., 2011), used with "uniform" and "most frequent" strategies. The "uniform" strategy generates prediction sampling with uniform probability from the set of labels that are provided during training. The "most frequent"

strategy always predicts the label that appears most frequently in the training data. The baselines were used as a comparison for the main focus of the research: BERT-based models. Different approaches were implemented, testing different architectures and configurations. The final model, shown in Figure 1, is thus composed of a BERT Encoding layer, which then splits into two classification heads. Each classification head is composed of a dropout layer and a linear layer. The two heads process the output independently, this leads to learning different features for the two different tasks. This approach was clearly better than using a single classification head and combining the classes. Several experiments were carried out trying to add LSTM layers, as proposed in (Lu et al., 2023) and in (Wang et al., 2022), but it only increased model complexity without improving the results. This finding was interesting, showing how the dataset had poor internal organization and unclear features to be learned by the models.

The experiments were carried out using a custom training and evaluation loop, inside which the loss function is computed. The optimal choice for the loss was a combination of CrossEntropy, used for emotions, with a BCEWithLogitsLoss, used for triggers. BCEWithLogitsLoss combines a Sigmoid layer and the BCELoss (Binary Cross Entropy) in one single class. This version is more numerically stable than using a plain Sigmoid followed by a BCELoss as, by combining the operations into one layer, we take advantage of the log-sum-exp trick for numerical stability (Paszke et al., 2019a). The two losses are combined inside a custom function to obtain the total average loss. The function also gives the possibility to balance the weight of each loss, by multiplying each loss value by a normalized scalar, obtaining a weighted average. The choice of PyTorch as neural library was determined by how convenient it makes the customization of the loss function.

3 Data

The dataset MELD (Poria et al., 2019a) is the one utilized, which is a transcript of the lines of the sitcom "Friends". The corpus consists of 4000 dialogues, which can be spoken by a minimum of 1 character to a maximum of 8. Each dialogue spans a range of lines, called "utterances" in the dataset, from 2 to 24. Each utterance is annotated with an emotion, from the aforementioned set, and a trigger

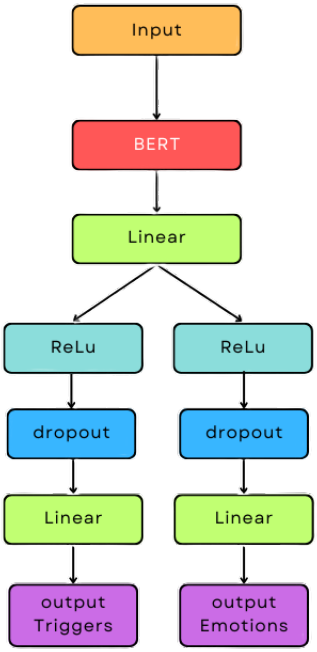

Figure 1: Model Architecture

binary label, signaling whether a sentence caused an emotion flip or not. However, it was evident that triggers were not consistently labeled: only the last emotion flip on each dialogue is annotated. This makes the trigger labeling criteria challenging to understand for humans since many dialogues are the continuation of another dialogue in the dataset. Exploring data made clear that strong class imbalance was present for emotions, as shown in Figure 2, as also for the triggers as in Figure 3. The data was split into training, testing, and validation sets with an 80/10/10 split.

Initial pre-processing of data included the imputation of Nan trigger values, replaced by a zero. Worth mentioning an attempt made to improve the trigger labeling. The idea is to check if an episode contains the same utterances of another episode and a few more, so meaning it is an extension of the previous, then the trigger annotations from the previous episode were replicated in the current episode. Even if this approach felt like improving the quality of data, it didn't improve significantly the metrics of any of the models trained. Then one-hot encoding was applied to the emotion labels. Finally, the trickiest part was designing a custom function that splits the utterances sentence by sen-

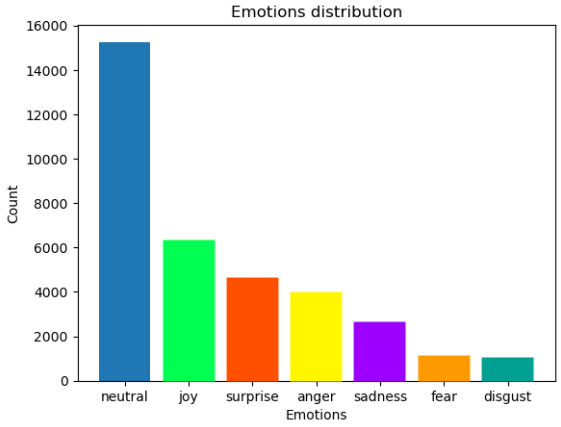

Figure 2: Emotions labels

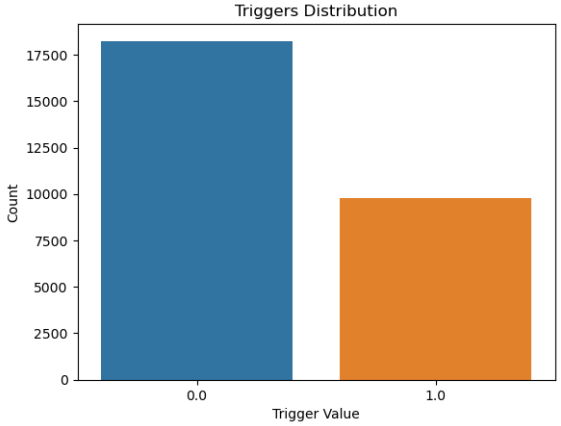

Figure 3: Triggers labels

tence and recombines them in inputs for the models. The function features a dynamic span window, that can be expanded up to the dialogue max length or shrunk down to a single utterance. It returns two new columns: "hist_curr" containing the current utterance with its history, i.e. all the sentences preceding it, and "next", containing all the next sentences. Both the history and the next utterances span over the size of the selected window. The concept was to create a dynamic environment, where it is possible to select how much of the whole dialogue to present to the model, allowing for different attention spans. The idea came by trying to expand the concept of "Attention" used by Transformers (Vaswani et al., 2023), by giving this feature also to the dataset, which could become, in a sense, "dynamic". After these pre-processing steps, tokenization was applied using BertTokenizer, which was applied taking into account the dynamic context window.

4 Experimental setup and results

To address the above-described tasks, several models were implemented and tested. Everything was implemented using a Python notebook, which was then run both on Google Colab and locally on GPUs, allowing a more flexible environment, especially for fine-tuning the model over hyperparameters. The baseline models were tested on the raw dataset, without tokenization, obtaining the results as shown in Table 1 for the emotion classification and the trigger classification. Then different BERT models were instantiated and tested, based on the same architecture.

The first model utilized is a frozen model, where only the classification heads are trained, obtaining relatively bad results, even after making substantial changes to the hyper-parameters. The second model tested was a fully trained model, which clearly demonstrated better performances. All the models were evaluated on the same metrics. Accuracy was computed both for the training set, during each epoch, and the validation set. The main evaluation metric was the F1 score, which was computed in two ways for each model: unrolled sequence and episode-based sequences. The unrolled F1 score evaluates the models by considering all utterances, regardless of which episode they belong to, considering an overall performance. The episode-based sequence F1 score is computed for the utterance of each dialogue, considering the context. The uniform average of the score for each dialogue is then reported. The F1 scores obtained with the 2 classifiers are shown in Table 2.

In order to work around the heavy class imbalance previously discussed classification weights for the emotion classification were introduced. The weight of each class c was calculated as $w_c = \sqrt{\frac{n\_zeros(c)}{n\_ones(c)}}$ where $n\_ones(c)$ is the number of utterances of the class c and $n\_zeroes(c)$ is the number of utterance that do not belong to the class c. The weights were further normalized to have sum equal to 1 in order to minimize the impact of the weighting on the learning rate. This gives the fairest comparison between the performance of the classifier with and without weights. Using the weights, again two different models were tested, one with the frozen transformer and one fully trainable. Using the weights the frozen model maintained the same level of performance. The full model was evidently disrupted by the class

c

Baseline	Majority		Random	
F1 score	AVG	σ	AVG	σ
Seq. Emo.	0.435	0.016	0.145	0.003
Seq. Tri.	0.646	0.014	0.501	0.013
Unr. Emo.	0.434	0.012	0.142	0.004
Unr. Tri.	0.647	0.012	0.503	0.010

Table 1: F1 Scores of the Baseline models

BERT	Full		Frozen	
F1 score	AVG	σ	AVG	σ
Seq Emo.	0.858	0.023	0.426	0.004
Seq Tri.	0.831	0.027	0.648	0.015
Unr Emo.	0.852	0.021	0.251	0.002
Unr Tri.	0.829	0.018	0.505	0.001

Table 2: F1 Scores of the BERT models

weighting, as shown in table 3. All the previous experiments were carried out with a window span of 4 utterances. It was determined that this setting is the best compromise of performance, in terms of information provided to the models, and memory allocation on the GPU. Conducting the experiments with a window span of 2 or lower is detrimental to the classification: there are not enough utterances to build a context upon. The above-mentioned experiments were all performed using Adam optimizer. In this context, further experiments were performed using AdaGRAD Optimizer, with a weight decay of 1e-3 and a Learning Rate Decay of 2e-7. Three epochs were performed for each experiment in order to allow the optimizer to fine-tune the weights and learning rate, and a slight increase in performance was achieved. For the reproducibility of the experiments, all the random seeds were manually fixed before each experiment. All the experiments with BERT were repeated with five different random seeds. The reported scores are the average of the results over five runs. The standard deviation of the F1 scores across the five executions is reported together with the F1 scores.

BERT+W	Full		Frozen	
F1 score	AVG	σ	AVG	σ
Seq Emo.	0.384	0.023	0.398	0.005
Seq Tri.	0.868	0.027	0.664	0.015
Unr Emo.	0.387	0.021	0.274	0.505
Unr Tri.	0.876	0.015	0.508	0.005

Table 3: F1 Scores of the Bert weighted models

5 Discussion

The two baseline models, The Random Classifier and The Majority Classifier serve as a benchmark for the other models. First of all, it was clear how the Majority baseline performed better than the Uniform counterpart. This is due to the data unbalance: since there is a class that is much more frequent than the others ('neutral' for the emotions and '0' for the triggers) choosing always the most frequent class is advantageous over a guess with uniform probability. The F1 score of the baseline models is higher for the classification of the triggers, being a binary classification with a high imbalance. The standard deviation of the majority is greater than zero since the data were shuffled at every iteration of the experiment.

The frozen BERT model performed only marginally better than the random classifier. On the other hand, the F1 scores of the majority classifier are similar to the one of the frozen BERT model. The model even performs worse than the baseline when we consider the overall performance over the whole dataset. A hypothesis on why this is the case is that the loss of the binary classification (BCE) has a higher magnitude than the one of the multiclass classification (CrossEntropy) on a misclassified sample. Therefore the classification head for the multiclass problem may not be sufficiently trained. This is supported by the fact that the used learning rate is relatively low: 5e-5. The loss function code in the notebook is already set up for balancing the loss of the emotions more than the loss of the triggers to perform the experiment to prove the hypothesis.

The full BERT model performed better than both the baselines and the frozen model. This shows how freezing the BERT layer is detrimental to performance, even if the training times are shortened by a factor of three. With this model the F1 score of the emotions is marginally better than the one of the triggers.

The frozen BERT model with weights performed worse than expected, showing that simply adding weights to the frozen model wasn't enough. Similar behavior was shown by the full BERT model with weights. It showed similar performances on trigger as the full BERT model not using weights, but performed clearly worse on emotions. It even reached lower F1 scores than the baselines. These two results show how weight computing wasn't optimal, so a possible future improvement could

consider another way to compute them.

6 Conclusion

The performed experiments showed different performances for four different BERT models. We compared them with a Random and a Majority classifier baseline. The best model was for sure the Full Bert model not using weights, as it showed clearly superior performance with respect to the others. The results also show how training only the classification heads is not enough to address the two tasks.

Another important finding was on using weights to address class imbalances, which did not work at all. Our work explored different ways of dealing with the two tasks, leading to interesting findings. The most important ones were on triggers replication and the dynamic context window, which has strong potential for different future experiments. However, the results obtained can surely be improved, by trying different techniques and with more computational resources available, as they were one of the biggest limitations encountered during the development of the discussed experiments.

Finally, this report clearly shows how data structure and organization are crucial to obtain good and satisfying results as often demonstrated in this field.

7 Links to external resources

All the materials of this report are available at the following GitHub repository

References

Menai Mohamed El Bachir Alswaidan, Nourah. 2020. A survey of state-of-the-art approaches for emotion recognition in text. *Knowledge and Information Systems*.

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

Wang L Chen M Yang D Li B Guo R, Guo H. 2024. Development and application of emotion recognition technology - a systematic literature review. *BMC psychology vol. 12,1 95*.

Guangyao Lu, Yuling Liu, Jie Wang, and Hongping Wu. 2023. Cnn-bilstm-attention: A multi-label neural classifier for short texts with a small set of labels. *Information Processing Management*, 60(3):103320.

The pandas development team. 2020. pandas-dev/pandas: Pandas.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019a. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019b. Pytorch: An imperative style, high-performance deep learning library.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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

Soujanya Poria, Navonil Majumder, Rada Mihalcea, and Eduard Hovy. 2019b. Emotion recognition in conversation: Research challenges, datasets, and recent advances.

Mahdi Rezapour. 2024. Emotion detection with transformers: A comparative study.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Shoujin Wang, Yuanjiao Yang, and Xin Meng. 2022. Research on multi-label text classification based on multi-channel cnn and bilstm. In *2022 International Conference on Artificial Intelligence of Things and Crowdsensing (AIoTCs)*, pages 498–503.