Title

Abstract

The task of emotion classification is overly researched nowadays including several different approaches and models. The large volume of different approaches makes it hard to understand what works and what is not anymore state of the art in its domain. In this study we implement and compare several methods for Multilabel emotion classification on the widely used dataset GoEmotions and evaluate their performance on the specific task. Additionally, we examine how different parameters such as schedulers, epochs and early stopping methods affect their performance and whether their adjustment improves the prediction of the least represented classes. Finally, we test how the selected architecture generalizes to a completely different dataset. The results show that external resources can boost the performance of the models as well as adjusting the early stopping method based on the task at hand. Finally, the different schedulers also made an impact on the model performance, which shows more research and experimentation needs to be made to finalize which scheduler best fits this problem.

Keywords

1. Introduction

Questions we tried to study

1) Is there any approach that clearly outperforms the rest in multilabel emotion recognition from the ones delivering state of the art results?
2) Does early stopping based on validation loss makes sense when the problem includes an imbalanced multilabel dataset?
3) How do schedulers affect the performance of the models
4) How well does the preferred model architecture generalize to other datasets?

2. Related Work

- Emotion classification using simple recurrent neural networks
- Knowledge enhanced text representations (early/late fusion)

3. Datasets and evaluation metrics

3.1. Datasets

Reference	Source	Language	Instances	Labels
Matta (2020) [1] – semEval2018	Twitter	English	~11K	11
Demszky et al. [2] - GoEmotions	Reddit	English	~54K	28

Table: Dataset information

Dataset	*GoEmotions*	*EC*
All	54K	10983
Train	43410	6838
Validation	5426	886
Test	5427	3259

Table 1: Train test split statistics

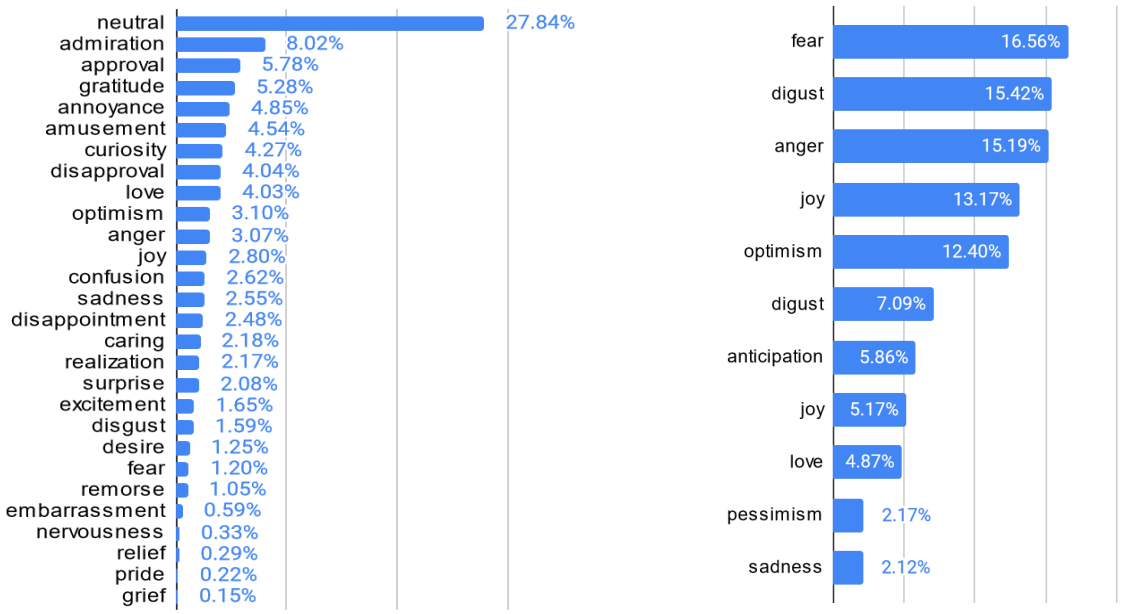


Figure 1: Class distribution in the two datasets, GoEmotions (left) and semEval-2018 (right)

3.2. Evaluation metrics

3.2.1. Jaccard similarity coefficient score

The Jaccard similarity coefficients is used to compute the similarity between pairs of label sets. The similarity coefficient is defined as:

$$J(y_i, \widehat{y_i}) = \frac{|y_i \cap \widehat{y_i}|}{|y_i \cup \widehat{y_i}|} \quad (1)$$

Where y_i is the true label of the i^{th} sample $\widehat{y_i}$ is the predicted label. In the case of Multilabel classification, such as our case, the Jaccard score of all samples is calculated by finding the metrics of each instance and then calculating their average.

3.2.2. Precision, Recall and F1 score

3.2.2.1. Precision

The precision is calculated using the equation (2), which is explained as the total correctly predicted positive instances divided with the total positive instances. High precision is preferred when there is no problem if there are some wrong predictions. Examples of such applications may be recommendation systems, suggesting the wrong video would not lead to an issue.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

3.2.2.2. Recall

The precision is calculated using the equation (3), which is explained as the total correctly predicted positive instances divided with the total real positive instances. High recall is important to models that having false negatives can cause problems. An application of such model would be predicting whether someone is having cancer or not. Having a large amount of false negatives could lead to misdiagnosis.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

3.2.2.3. F1 score

F1 score takes into account both precision and recall, it is basically their harmonic mean, therefore shows how the model performs in a more balanced way. High F1 score shows that the model performs well in both recall and precision.

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

3.2.2.4. Aggregation

Each of these three metrics can be calculated in many different ways if the classification problem is not binary. There are several aggregation methods for multiclass and Multilabel classification problems which are described as followed:

- Micro Average: This computes the F1 score considering the total true positives, false negatives and false positives.
- Macro average: Calculates the F1 score per class and takes their average.
- Weighted average: Calculates the F1 score per class and takes the weighted average, meaning it considers the proportion of the dataset of each label.
- Sample average: Calculates the F1 score for each instance and then it returns the average.

For Multilabel classification, all these aggregation methods can be performed, but macro average is the one that will give a more general overview of the model's performance and is more widely used for evaluation of such tasks.

4. Methods

A large amount of the methods that achieve state-of-the-art performance in downstream NLP tasks and specifically in the emotion classification task, heavily reside on transfer learning and on the usage of pretrained models. These pre-trained models provide text representations which includes information regarding all surrounding words and the general context of the text. Therefore, all of our proposed architectures use as input embedding the output of

the pretrained transformer models, which is in short, the contextual representation of the text which ultimately contains meaningful information. More specifically we use the pretrained BERT model (bert-base-uncased) which was first published by Devlin et al. [3] in 2019 and is publicly available.

In this section we first describe a baseline method which simply uses the contextual representation of the sentences. This is followed by two more advanced model architectures which both use the contextual representation of the BERT model and then, the first incorporates LSTM and BiLSTM architectures and the second includes external emotion related resources and a self-attention mechanism. All architectures treat the multilabel problem as multiple binary classification problems, where each binary classifier outputs a probability which then becomes positive or negative based on a threshold of 0.3.

4.1. BERT Fine-tuning with a fully connected layer

To set the baseline performance, we first experiment with a similar architecture to the one presented by Demszky et al. [2] in when they first published the GoEmotions dataset. We pass the tokenized input through the pretrained BERT model and we extract the last hidden state of the classification token – the contextual representation of the input text. This is then passed through a fully connected layer with results to a tensor of size 50 and then through an activation layer using ReLU. Finally, dropout is performed, and it is final passed through a fully connected layer resulting to a tensor of size 28 containing the logits. The logits become probabilities using a simple sigmoid layer. (Figure 2 - left)

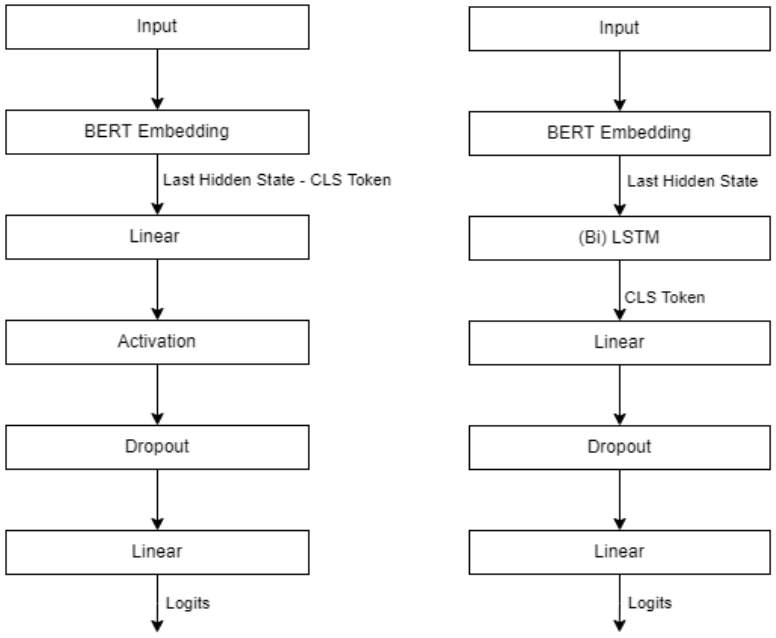


Figure 2: Model architectures, BERT fine-tuning (left) and BERT LSTM/BiLSTM (right)

4.2. Stacked LSTM and Bi-LSTM layers

Neural architectures have the following properties - they can be single direction or bidirectional. More recent adjustment are the LSTMs. – add the advantage to the contextual repr of the input text of the properties of the bilstm models (extra memory??)

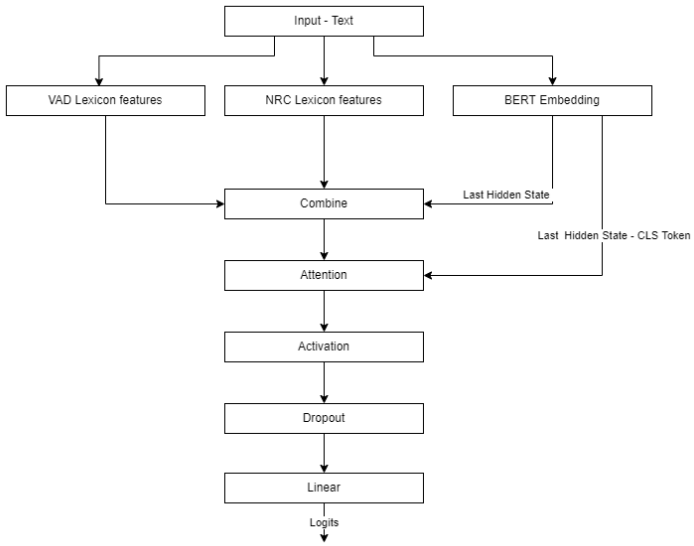
Therefore, using RNN models, such as LSTM and BiLSTMs, is quite common and in the recent bibliography they perform adequately to be considered as state of the art for such tasks.

In our implementation we experimented using the last hidden state of the output of the pretrained models and passing it though several different RNN layers such as LSTM and BiLSTM with different parameter settings. The most promising results were given using an BiLSTM layer with 200 hidden states. The results of the LSTM layer are presented in the section XX, but it was observed that the output of this layer was not consistent while using different random seeds. A dropout of 0.3 was used as well as two fully connected layers. (Figure 2 – right)

4.3. External resource boosting with self-attention mechanism

In the past, it was common approach for emotion classification tasks to use external resources that encode the emotional knowledge such as emotion lexicons. These resources provided the necessary information connected to specific emotional words to create extra features that a simple machine learning model such as SVM could benefit from, to increase its performance in this specific task. Even though these types of models lacked the contextual information which limited their performance, the idea of incorporating external knowledge was promising. Therefore, in this model a combination of the contextual representation of the text as well as the emotional knowledge of two different lexicons is made in order to boost the performance of the model. The combination of these three aspects is done similarly to the Sentence-level KEA presented by V. Suresh and D. C. Ong [4] and it is presented in the .

In short, the emotion knowledge of the two lexicons is transforming the input text to two feature vectors which are incorporating the information of the lexicons. These feature vectors are fed to a fully connected layer and their output (vad, nrc) is concatenated with the final hidden state of the classification token (h_0) to create the Key K of an attention mechanism (equation **Error! Reference source not found.**. This attention mechanism is using as the Query the final hidden state of the pretrained model, the Key and the Query are multiplied to provide the attention score (equation **Error! Reference source not found.**). The softmax of the attention score is multiplied with the Key to output the final representation of the input text ($h_{0_{new}}$) (equation **Error! Reference source not found.**). This is then passed through fully connected layers as well as a dropout layer to finally output the logits used for the classification task.



$K = concat(vad, nrc, h_0)$	*(5)*
$s = softmax(h_0^T * K)$	*(6)*
$h_{0_{new}} = s^T * K$	*(7)*

Figure 3: Model architecture using external resource boost

4.4. Weighted Loss

Considering we are facing the imbalanced dataset challenge a known method to tackle this issue is by using an adjusted loss that considers the imbalance. Our approach includes the calculation of weights for all emotion classes and then the calculation of a weighted loss. As mentioned earlier, the approach followed for the multilabel issue in our experiments, with 28 different emotion classes, is by considering twenty-eight different binary problems. So with C_i we describe one emotion class (eg. $C_{happiness}$), where i each one of the 28 emotion. In other words, with C_i we describe one binary classification problem which has two possible labels, zero and one. This roughly translates whether there are signs of happiness in the text (label one – or positive) or not (label zero – or negative).

For each binary emotion problem two weights are calculated one for the positive class and one for the negative. The equation used for this calculation is as followed.

$$W_{i,0}, W_{i,1} = \frac{n_{samples}}{n_{classes} * [n_{pos,} n_{neg}]} \quad (8)$$

Where $n_{samples}$ is the total of all instances of the dataset, $n_{classes}$ is two since we face a binary problem, n_{pos} is the amount of positive samples and n_{neg} the amount of negative samples. The outputs of this are the weights of the positive and negative samples of one emotion class which basically describes the imbalance of that emotion class.

For the calculation of the weights loss, these weights are taken into account in a combination of the calculation of the loss with a normal loss function. In more detail the weighted loss calculation is shown in equation X and for one example instances it roughly translates to the normal loss of that instance multiplied by the weights of the true label. In our case as $loss\_fn$ we use the binary cross entropy loss.

$$Weighted\ loss = AVG(w_0^{(1-y_{true})} * w_1^{y_{true}} * loss\_fn(logits, y_{true}) \quad (9)$$

Therefore all of our experiments were performed both by using and without using the weighted loss. Another known approach to improve performance in the multilabel challenge is by using different thresholds for each binary classification problem. Such approach was also used and tried upon, but did not result to any useful outcomes.

5. Results & Discussion

5.1. Experimental setup

Each experiment was repeated five times with different random seeds, and we report the mean and the standard deviation of the performance over all five runs. For running the experiments, we used an NVIDIA GeForce RTX 3060 GPU.

The input text before fine-tuning of all our models has been pre-processed by converting all characters to utf-8 encoding, by converting all letters to lower case, transforming the emojis to their respective words, removing the contractions, the hyperlinks, the "#" character from the hashtags, and the non-necessary white-spaces. Then all text

was tokenized, padded or truncated to maximum length which in our case was 128 tokens and added the special tokens '[CLS]' and '[SEP]'.

For fine-tuning BERT initially all experiments were performed using the parameters presented in the literature, which can be found in table X. This approach was followed in order to be comparable with the state-of-the-art published.

Parameter	**Value**
Optimizer	AdamW (eps=1e-8, β1= 0.9, β2=0.99)
Learning rate	5e-5
Scheduler	Linear with warmup[1]
Dropout	0.3
Early stopping metric	Validation loss

Table 2: Initial experiment parameters

After the initial experiments, the best performing model of ours was used to fine-tune the aforementioned parameters. This was in order to identify whether there are any improvements that can be made that the current state of the art publications have not yet identified.

5.2. Preliminary results with literature hyper parameters

The comparative results of all models mentioned in section X can be found in table X. For our analysis we mostly pay attention to the weighted F1, since it is the one that can mirror best how well the model is performing regardless of the imbalance of the dataset, in the multilabel classification problem. As anyone can see, the weighted loss is improving the performance of all models when it is used. Nevertheless, we can identify that the usage of external resources indeed gives a performance increase by 3% compared to the baseline model with the weighted loss usage, and a 5% increase compared to the baseline model without the weighted loss.

Model	**Jaccard**	**Macro - F1**	**Weighted - F1**	**Samples - F1**	**Recall**	**Precision**
BERT	0.55658 (0.0122)	0.4435 (0.0159)	0.5760 (0.0089)	0.5880 (0.0130)	0.5300 (0.0100)	0.4180 (0.0164)
BERT - LSTM	0.4660 (0.1045)	0.2718 (0.2041)	0.4120 (0.1942)	0.4900 (0.1202)	0.3020 (0.2364)	0.2720 (0.1906)
BERT - BiLSTM	0.5669 (0.0053)	0.4608 (0.0082)	0.5860 (0.0055)	0.6060 (0.0055)	0.5020 (0.0084)	0.4560 (0.0055)
BERT - VAD - NRC	0.5719 (0.0042)	0.4800 (0.0098)	0.5940 (0.0055)	**0.6120 (0.0084)**	0.5180 (0.0148)	0.4800 (0.0122)
BERT – WL	0.5538 (0.0109)	0.4645 (0.0145)	0.5780 (0.0045)	0.5860 (0.0114)	0.5400 (0.0122)	0.4440 (0.0207)
BERT - LSTM - WL	0.5089 (0.5089)	0.3689 (0.3689)	0.4880 (0.1946)	0.5380 (0.1221)	0.4060 (0.2223)	0.3640 (0.1820)
BERT - BiLSTM - WL	0.5606 (0.0080)	0.46454 (0.0087)	0.5800 (0.0071)	0.5980 (0.0084)	0.5040 (0.0270)	0.4600 (0.0100)
BERT - VAD - NRC - WL	**0.5727 (0.0052)**	**0.4938 (0.0057)**	**0.5960 (0.0055)**	**0.6120 (0.0045)**	**0.5420 (0.0084)**	**0.4900 (0.0071)**

[1] Transformers function: https://huggingface.co/docs/transformers/main_classes/optimizer_schedules#transformers.get_linear_schedule_with_warmup

Table 3: Experiment results of all model architectures using the GoEmotions dataset. The results are an average of 5 repeats using different seed with their standard shown in parenthesis

The performance of the state-of-the-art implementations, which are found in the literature featuring the GoEmotions dataset, are shown in Table 4, as well as our experimental results. For better comparison the state-of-the-art implementations having the best results[2] are also executed in our setting in order to verify or contradict the published results.

The published results can be found in the first part of the table as well as the reproduced outcomes in the second part. In the third part our results are presented for easier comparison. It can be seen that the published results are better than the ones that we got after re execution of their implementation. This may be due to different setting, or different seeds used for initialization of the models among other reasons. Nevertheless, even with the reproduced outcomes, our results are slightly lower than the rest.

	Model	Macro - F1	Recall	Precision
Published outcomes	BERT base [2]	0.46	0.63	0.4
	BERT DNN pool [5]	0.48	0.52	0.61
	KEA – BERTsentence [4]	0.51 (0.007)	0.525 (0.006)	0.514 (0.022)
	BERT – MLM [6][3]	0.5125	0.5227	0.5242
	BERT – CDP [6]3[3]	0.5234	0.538	0.5466
	BERT – MLM – CDP [6][3]	0.5196	0.5437	0.5342
Reproduced outcomes	KEA – BERTsentence [4]	0.5008	0.5013	0.5265
	BERT – MLM [6][3]	0.5033	0.5253	0.5050
	BERT – CDP [6] [3]	*Cannot reproduce – no config file*		
	BERT – MLM – CDP [6][3]	0.4976	0.5207	0.5153
Our implementation	BERT - WL	0.4645 (0.0145)	0.5400 (0.0122)	0.4440 (0.0207)
	BERT - LSTM - WL	0.3689 (0.3689)	0.4060 (0.2223)	0.3640 (0.1820)
	BERT - BiLSTM - WL	0.4645 (0.0087)	0.5040 (0.0270)	0.4600 (0.0100)
	BERT - Vad - NRC - WL	0.4938 (0.0057)	0.5420 (0.0084)	0.4900 (0.0071)

Table 4: Comparative view of our the best performing model with the state-of-the-art reported results

5.3. Optimizing the Macro F1 score

As stated before, so far, our implementations even though they are guided by the ideas existing in the literature, they still don't reach their performance. So, in order to increase our performance and the macro F1 results, several tweaks were made to our models, which are described in the following sections.

5.3.1. Early stopping method

As shown in Table 2, our implementation is following the standard early stopping metric, validation loss as in a lot of research activities to avoid overfitting. Nevertheless, in such problems having imbalanced dataset in a multilabel task it is possible for the validation loss to increase while the actual metric we need to observe also increases, something that can be observed also in our case (

[2] Only for the ones that have published code

[3] [6] Results are all with a different train-dev-test split than the original presented in [2], maybe the results are not comparable

Epoch	Jaccard	M-F1	Loss
1	0.559	0.4563	0.088017
2	0.5772	0.4939	0.085892
3	0.5478	0.4982	0.091291
4	0.5386	0.5007	0.101981
5	0.5474	0.5124	0.119644
6	0.5437	0.485	0.132933
7	0.5493	0.4994	0.147578
8	0.538	**0.5137**	0.156212
9	0.5431	0.5086	0.164596
10	0.5422	0.501	0.16941

Table 5). Therefore, in order to optimize the metric at hand, meaning Macro F1 score, we opted out the usual Early Stopping metric (validation loss) and decided to use the macro F1 score instead.

Epoch	Jaccard	M-F1	Loss
1	0.559	0.4563	0.088017
2	0.5772	0.4939	0.085892
3	0.5478	0.4982	0.091291
4	0.5386	0.5007	0.101981
5	0.5474	0.5124	0.119644
6	0.5437	0.485	0.132933
7	0.5493	0.4994	0.147578
8	0.538	**0.5137**	0.156212
9	0.5431	0.5086	0.164596
10	0.5422	0.501	0.16941

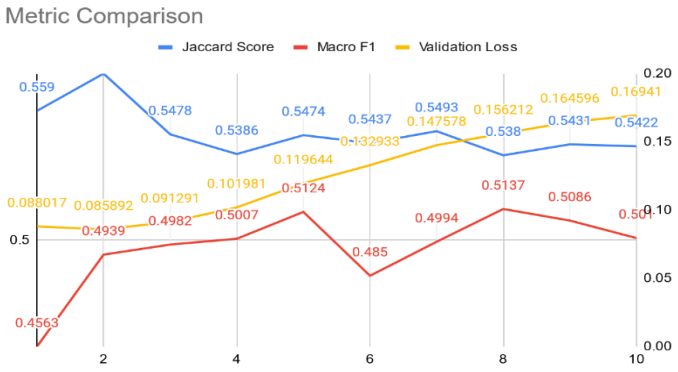


Table 5: Jaccard, macro-F1 and validation loss of the test set for 10 consecutive epochs. This shows that the best macro-F1 for the test set is not when the validation loss is minimized.

5.3.2. Scheduler change

An additional adjustment that we experimented with is the usage of different schedulers. Adding to the original scheduler, “linear schedule with warmup” we experimented with (a) a combination of two schedulers, a linear and a step scheduler, now on called “chained schedulers” as well as (b) an “adjusted cosine annealing with decay” scheduler. How the learning rate changes per epoch and training step can be seen in Figure X, for all three different schedulers.

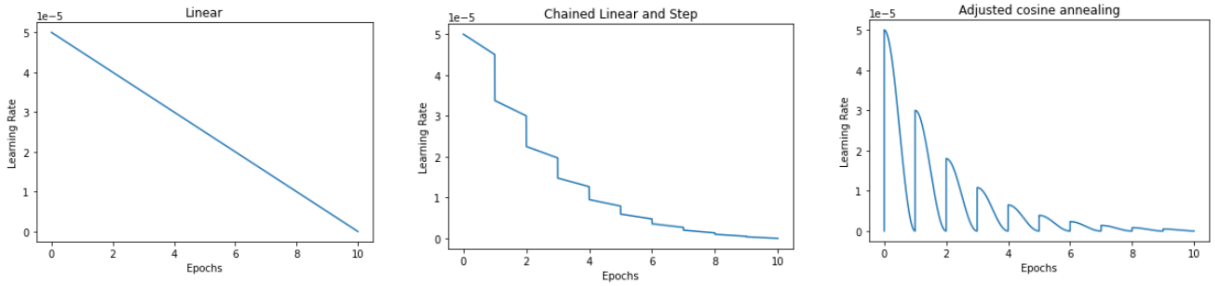


Figure 4: The learning rate (y axis) change per epoch (x axis) of the different schedulers. (a) linear schedule with warmup, (b) chained schedulers, (c) adjusted cosine annealing with warm restarts

With the same model parameters experiments using the BERT – VAD – NRC model were executed with the three different schedulers. The test macro-F1 was calculated at the end of each epoch for the range of ten epochs to identify whether all schedulers had the same epoch the best performance for the macro F1 as well as to identify which scheduler better fits the emotion classification multilabel problem. The chained scheduler has the best performance between all three when checking the validation set (Table 6). Performing the same process on the test set (*Table 7*), it shows that the validation and test best performing epoch regarding the macro F1 score are not aligned. This shows that different information are present in the two sets.

Epoch	Linear	Chained	Cosine
1	0.4511	0.4511	0.4214
2	0.4915	0.4882	0.489
3	0.5003	0.5054	**0.5079**
4	**0.5083**	0.5149	0.5068
5	0.4906	**0.5115**	0.4935
6	0.4900	0.5086	0.4897
7	0.4899	0.5076	0.4905
8	0.4806	0.5055	0.4918
9	0.4945	0.5049	0.4930
10	0.4929	0.5022	0.4928

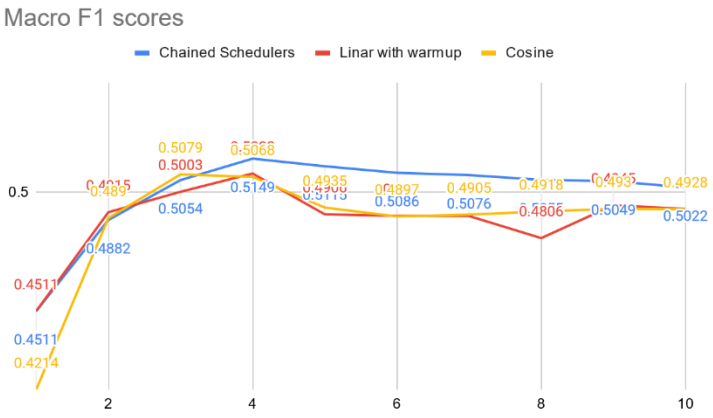


*Table 6: Macro F1 score of the **validation** set at the end of ten consecutive epochs for the three suggested schedulers*

Epoch	Linear	Chained	Cosine
1	0.4563	0.4563	0.4254
2	0.4939	0.4988	0.4886
3	0.4982	0.5073	0.5134
4	0.5007	0.5068	0.515
5	0.5124	**0.512**	0.5179
6	0.4850	0.4972	0.5281
7	0.4994	0.5011	0.5236
8	**0.5137**	0.5063	0.5197
9	0.5086	0.5043	**0.5284**
10	0.501	0.5094	0.527

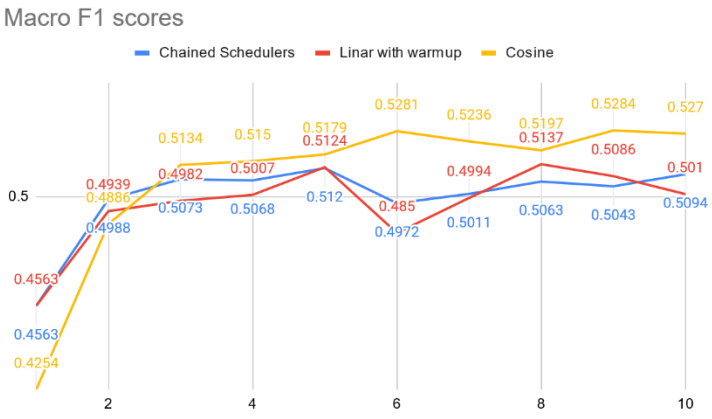


Table 7: Macro F1 score of the test set at the end of ten consecutive epochs for the three suggested schedulers

5.3.3. Usage of sparsemax

Softmax is a function that is was first introduced by Bridle in 1989 [7] and it is key component on several machine learning models, sometimes used to transform the model outputs to probabilities. One characteristic of Softmax is that the resulting probabilities are different larger than zero for all values of the input vector. That means Softmax cannot explain well enough sparse distributions, to do so, sometimes a threshold is defined bellower which the probabilities are made to zero. An alternative function that can model sparse distributions is the Sparsemax transformation.

The Sparsemax transformation, first introduced by Martins and Astudillo [8], has the ability to return the sparse posterior distributions of an output, simply that means it can return zero probability to some output variables. This ability can be used for several applications such as predicting multiple labels, rather than just one, or to identify which variables have larger impact on a potential decision among others.

In our case we suggest using sparsemax[4] instead of softmax in the computation of the attention layer, which can potentially show which variables of the hidden state are more important, and zero out the ones that are not important at all. In more detail, the equation y becomes:

$$s = sparsemax(h_0^T * K) \quad (10)$$

The other parameters such as dropout and the initial learning rate used in the original set of experiments were found to be the better choice after a set of fine-tuning experiments.

5.3.4. Results

With this adjusted Early Stopping method, the different scheduler and the usage of sparsemax further experimentation was performed, all using the BERT-VAD-NRC model which was the one performing best in the original experiments. Results of these additional experiments can be found in Table 8.

Model	ES Method	Scheduler	Attention method	Macro - F1	Recall	Precision	Grief F1	Relief F1	Epoch
	Validation Loss	Linear	Softmax	0.4938 (0.0057)	0.5420 (0.0084)	0.4900 (0.0071)	0.0000 (0.0000)	0.0000 (0.0000)	2

[4] Sparsemax library public repository: https://github.com/aced125/sparsemax

BERT - Vad - NRC - WL	Macro F1	Linear	Softmax	0.5053 (0.0087)	0.4980 (0.0192)	0.5320 (0.0084)	**0.2460 (0.2249)**	0.3020 (0.0779)	4
	Macro F1	Linear	Sparsemax	0.5121 (0.0102)	0.5140 (0.0195)	0.5300 (0.0071)	0.2320 (0.2361)	0.3280 (0.1211)	3.6
	Macro F1	Cosine	Softmax	0.5061 (0.0047)	0.5140 (0.0114)	0.5240 (0.0055)	0.0000 (0.0000)	0.2460 (0.1804)	5.6
	Macro F1	Cosine	Sparsemax	0.5078 (0.0060)	0.5140 (0.0152)	0.5240 (0.0055)	0.0000 (0.0000)	0.1740 (0.1839)	5.2
	Macro F1	Chained	Softmax	**0.5133 (0.0079)**	**0.5260 (0.0167)**	0.5300 (0.0071)	0.1580 (0.2287)	0.2880 (0.0789)	3.6
	Macro F1	Chained	Sparsemax	0.5125 (0.0150)	0.5120 (0.0277)	**0.5340 (0.0114)**	0.1160 (0.1664)	**0.3880 (0.1346)**	4.4

Table 8: Results of all experiments with combinations of the suggested improvements – scheduler and attention method

From this analysis there are some conclusions that can be made which are summarized below:

- Early stopping using Macro F1 as the early stopping metric indeed can improve how well the model is trained and the results on the test set
- The adjusted cosine does not performed as well as it would be expected while looking at *Table 7*. This is thought to be expected while using Validation macro-F1 as early stopping metric. As shown in Table 6, the macro f1 of the validation set maximizes in an earlier epoch than the macro f1 of the test set.

While looking at the individual categories and the performance of the different models/hyper parameters, we see an increase on the least represented categories (embarrassment, grief, nervousness, pride, relief) compared to the baseline model (Table 9). The earlier conclusion of the adjusted cosine scheduler performing worse than the other two is also supported here, which does not predict correctly at all the category "grief". All other suggested adjustments perform better in these under-represented categories, but there is no one stands out for all of them. Nevertheless, The combination of the chained scheduler with a softmax function in the attention mechanism gives better results in sixteen out of twenty-seven categories, whereas the next best combination gives the best results to only nine categories, out of which on two they both have the same result.

Emotions	**BERT - WL**	**BERT – VAD – NRC - WL**					
	baseline	**Linear**		**Cosine**		**Chained**	
		softmax	**sparsemax**	**softmax**	**sparsemax**	**softmax**	**sparsemax**
admiration	0.684	0.668	0.680	0.692	0.688	0.684	0.676
amusement	0.810	0.812	0.816	0.816	0.822	0.816	0.816
anger	0.476	0.478	0.484	0.500	0.498	0.502	0.502
annoyance	0.346	0.358	0.340	0.362	0.368	0.364	0.360
approval	0.368	0.364	0.354	0.382	0.390	0.384	0.376
caring	0.400	0.416	0.410	0.438	0.422	0.430	0.416
confusion	0.422	0.420	0.430	0.418	0.448	0.426	0.422
curiosity	0.532	0.530	0.534	0.560	0.564	0.560	0.552
desire	0.456	0.486	0.484	0.500	0.518	0.500	0.518
disappointment	0.282	0.290	0.292	0.328	0.332	0.312	0.322
disapproval	0.358	0.388	0.404	0.398	0.404	0.400	0.396
disgust	0.486	0.496	0.492	0.502	0.508	0.496	0.492
embarrassment	0.446	**0.486**	**0.486**	**0.482**	**0.488**	**0.490**	**0.466**
excitement	0.432	0.430	0.422	0.430	0.428	0.424	0.420
fear	0.650	0.638	0.654	0.670	0.664	0.660	0.664
gratitude	0.906	0.908	0.914	0.914	0.916	0.914	0.906
grief	0.058	**0.246**	**0.276**	**0.000**	**0.000**	**0.158**	**0.116**

joy	0.596	0.608	0.608	0.608	0.618	0.612	0.614
love	0.792	0.790	0.796	0.800	0.808	0.798	0.788
nervousness	0.388	**0.396**	**0.396**	**0.404**	**0.414**	**0.410**	**0.422**
optimism	0.550	0.538	0.548	0.560	0.564	0.564	0.560
pride	0.446	**0.448**	**0.502**	**0.476**	**0.498**	**0.470**	**0.490**
realization	0.222	0.236	0.246	0.236	0.240	0.256	0.240
relief	0.030	**0.356**	**0.328**	**0.246**	**0.174**	**0.288**	**0.388**
remorse	0.660	0.678	0.674	0.680	0.676	0.676	0.678
sadness	0.550	0.554	0.562	0.552	0.548	0.562	0.550
surprise	0.546	0.550	0.556	0.558	0.562	0.556	0.546
neutral	0.658	0.644	0.652	0.664	0.662	0.662	0.660
avg	0.484	0.5053	0.5121	0.5061	0.5078	**0.5133**	0.5125
std	0.203	0.166	0.168	0.192	0.195	0.175	0.174

Table 9: Comparison of Macro-F1 scores of each emotion category. The underlined-bold categories have less training support. The highlighted cells with stronger blue background are the best results in each category, the comparison is not taking into account the experiments using the cosine scheduler.

5.3.5. Evaluation of model architecture on a new dataset

Use all methods on semEval 2018 E-C and report the results

	Model			Macro - F1	Recall	Precision
Published outcomes	KEA – ELECTAsentence [4]			**0.591**	0.619	0.577
	KEA – BERTsentence [4]			0.582	0.613	0.568
	NTUA – SLP [9]			0.530	-	-
	TCS Research [10]			0.530	-	-
	LEM [11]			0.560	-	-
	BNet [12]			0.560	-	-
	Seq2Emo [13]			0.519	-	-
	referEmo [14]			0.570	-	-
Our implementation	BERT			0.5739 (0.0040)	0.5875 (0.0096)	0.5800 (0.0082)
	BERT - LSTM			0.2718 (0,2041)	0.3020 (0.2364)	0.2720 (0.1906)
	BERT - BiLSTM			0.5725 (0.0075)	0.5740 (0,0114)	0.5840 (0.0152)
	BERT - VAD - NRC			0.5721 (0.0078)	0.5700 (0.0122)	0.5920 (0.0148)
	BERT - WL			0.5717 (0.0049)	0.5840 (0.0114)	0.5820 (0.0164)
	BERT - LSTM - WL			0.5648 (0.0062)	0.5720 (0.0045)	0.5720 (0.0130)
	BERT - BiLSTM - WL			0.5745 (0.0086)	0.5700 (0.0100)	0.5940 (0.0241)
	BERT - Vad - NRC - WL			0.5770 (0.0096)	0.5720 (0.0148)	0.5980 (0.0277)
	VAD-NRC	Linear	Softmax	0.5770 (0.0096)	0.5720 (0.0148)	0.5980 (0.0277)
	VAD-NRC	Linear	Sparsemax	0.5733 (0.0106)	0.5775 (0.0206)	0.5925 (0.320)
	VAD-NRC	Cosine	Softmax	0.5804 (0.0041)	0.5820 (0.0045)	0.5980 (0.0045)
	VAD-NRC	Cosine	Sparsemax	0.5783 (0.0050)	0.5780 (0.0084)	0.5920 (0.0045)
	VAD-NRC	Chained	Softmax	**0.5829 (0.0064)**	0.5740 (0.0055)	0.6060 (0.0182)
	VAD-NRC	Chained	Sparsemax	0.5821 (0.0058)	0.5760 (0.0114)	0.6000 (0.0200)
	VAD-NRC	Linear	Softmax	0.5770 (0.0096)	0.5720 (0.0148)	0.5980 (0.0277)
	VAD-NRC	Linear	Sparsemax	0.5733 (0.0106)	0.5775 (0.0206)	0.5925 (0.320)

Table 10: Results of the different model architectures different schedulers, attention methods and the state-of-the-art results using the semEval-2018 dataset

From these results, we see that indeed, the conclusions made by the experiments on the GoEmotions dataset, generalize to the other dataset as well. Similarly to the GoEmotions, the semEval-2018 dataset gives the best macro-F1 when using a chained scheduler with a softmax attention mechanism.

6. Conclusions

Further work:

- Use label correlation techniques? – some labels have relations between each other
- Use newer language models like bloom?
-

7. Bibliography

[1] N. Matta, "Covid Twitter Emotion Analysis Data," vol. 1, Jul. 2020, doi: 10.17632/47hy8yyky5.1.

[2] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "GoEmotions: A Dataset of Fine-Grained Emotions," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, Jul. 2020, pp. 4040–4054. doi: 10.18653/v1/2020.acl-main.372.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *ArXiv181004805 Cs*, May 2019, Accessed: Nov. 15, 2021. [Online]. Available: http://arxiv.org/abs/1810.04805

[4] V. Suresh and D. C. Ong, "Using Knowledge-Embedded Attention to Augment Pre-trained Language Models for Fine-Grained Emotion Recognition," *ArXiv210800194 Cs*, Jul. 2021, Accessed: Mar. 17, 2022. [Online]. Available: http://arxiv.org/abs/2108.00194

[5] N. Alvarez-Gonzalez, A. Kaltenbrunner, and V. Gómez, "Uncovering the Limits of Text-based Emotion Detection," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, Punta Cana, Dominican Republic, 2021, pp. 2560–2583. doi: 10.18653/v1/2021.findings-emnlp.219.

[6] G. Singh, D. Brahma, P. Rai, and A. Modi, "Fine-Grained Emotion Prediction by Modeling Emotion Definitions." arXiv, Jul. 26, 2021. Accessed: Aug. 30, 2022. [Online]. Available: http://arxiv.org/abs/2107.12135

[7] J. Bridle, "Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters," in *Advances in Neural Information Processing Systems*, 1989, vol. 2. Accessed: Oct. 04, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/1989/hash/0336dcbab05b9d5ad24f4333c7658a0e-Abstract.html

[8] A. Martins and R. Astudillo, "From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification," in *Proceedings of The 33rd International Conference on Machine Learning*, Jun. 2016, pp. 1614–1623. Accessed: Sep. 29, 2022. [Online]. Available: https://proceedings.mlr.press/v48/martins16.html

[9] C. Baziotis *et al.*, "NTUA-SLP at SemEval-2018 Task 1: Predicting Affective Content in Tweets with Deep Attentive RNNs and Transfer Learning," *ArXiv180406658 Cs*, Apr. 2018, Accessed: Dec. 08, 2021. [Online]. Available: http://arxiv.org/abs/1804.06658

[10] H. Meisheri and L. Dey, "TCS Research at SemEval-2018 Task 1: Learning Robust Representations using Multi-Attention Architecture," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana, Jun. 2018, pp. 291–299. doi: 10.18653/v1/S18-1043.

[11] H. Fei, Y. Zhang, Y. Ren, and D. Ji, “Latent Emotion Memory for Multi-Label Emotion Classification,” *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 05, Art. no. 05, Apr. 2020, doi: 10.1609/aaai.v34i05.6271.
[12] M. Jabreel and A. Moreno, “A Deep Learning-Based Approach for Multi-Label Emotion Classification in Tweets,” *Appl. Sci.*, vol. 9, no. 6, Art. no. 6, Jan. 2019, doi: 10.3390/app9061123.
[13] C. Huang, A. Trabelsi, X. Qin, N. Farruque, L. Mou, and O. Zaïane, “Seq2Emo: A Sequence to Multi-Label Emotion Classification Model,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online, 2021, pp. 4717–4724. doi: 10.18653/v1/2021.naacl-main.375.
[14] A. Esperanca and X. Luo, “ReferEmo: A Referential Quasi-multimodal Model for Multilabel Emotion Classification,” in *Database and Expert Systems Applications*, Cham, 2022, pp. 351–366. doi: 10.1007/978-3-031-12423-5_27.