Delving into Fine-Grained Emotion Classification

Yuanxin Wang, Yiting Feng

Abstract

Emotion detection algorithms are widely adopted in various NLP applications including conversational AI and recommendation systems. We attempted to tackle the problem of fine-grained emotion detection, especially the GoEmotion dataset. We explored three directions to improve the performance of predicting the emotion in sentences, including joint training with auxiliary task, explicit attending contexts with label embedding, and embedding mixing. We also tested the potential to integrating the three method, with the best model achieving a relative improvement of 14% in the macro f1-score over the original baseline.[1]

1 Introduction

Emotion detection are essential to social interaction. There has been a handful of work on emotion expression and detection. Early work on emotion detection such as Affective Text (Strapparava and Mihalcea, 2007) modeled emotions with six basic emotions: anger, disgust, fear, joy, sadness and surprise, following Ekman's model (Ekman, 1992). Following-up works introduced both manually-constructed (Bostan and Klinger, 2018) and weakly-labeled emotion datasets (Wang et al., 2012). The most recent benchmark GoEmotions (Demszky et al., 2020) includes the largest human-annotated dataset, with multiple annotations per example, which is the benchmark tested in our work.

Our project focuses on fine-grained emotion detection. Our goal is to design an algorithm that predict the emotion expressed in input sentences. We started from a baseline BERT model (Devlin et al., 2019) and explored three directions, including adding auxiliary task, attending with label embeddings and embedding mixing.

[1]Data and code available at:
https://github.com/MichaelYxWang/11747-Project-GoEmo

We selected the auxiliary tasks based on the hierarchical relation between fine-grained labels and coarse labels like sentence sentiments (§3.2.2). We jointly trained the main task with those auxiliary tasks and observed improvements in performance (§3.2.1). We also extended the definition of existing information-theoretic measures and evaluated the similarity of our auxiliary tasks and the main task (§3.2.3).

Inspired by (Alhuzali and Ananiadou, 2021), we leveraged the rich semantic information in the pre-trained embedding of the emotion labels. We applied the attention mechanism between the emotion labels and the input documents, which leads to a large improvement in our model's performance. We explore interpretation to this improvement by visualizing the attention distribution (§5.2).

With the prevalence of the use of paraphrasing models (Raffel et al., 2019) and external knowledge (Khandelwal et al., 2020) in various NLP tasks including question answering and neural language modeling, we introduced controlled noise and training knowledge in our inference process. This approach results in a moderate improvement in performance and some future efforts in this direction is discussed in the later sections (§3.4).

2 Related Work

2.1 Emotion Classification Models

Both feature-based and neural models have been adopted to computationally model emotion and automate emotion classification.

Lexicon-based Methods Using the categorical basic emotion model(Plutchik, 1984), the emotion classification task benefits from early manually-crafted lexicons of valence, arousal, and dominance (Bradley and Lang, 1999; Warriner et al., 2013). The reliability of those lexicons is further improved by fixing the consistency of annotations (Mohammad, 2018), which further assisted the emotion detection task.

Early Neural-based Methods Early neural-based methods found that LSTM-based neural network could capture sequence information and assist emotion classification (Hochreiter and Schmidhuber, 1997; Lin et al., 2020). Neural-based methods benefits from the representation from large pre-trained transformer-based models like BERT (Devlin et al., 2019) which reached state-of-the-art performance on various NLP tasks including emotion classification: (Demszky et al., 2020) found that BERT-based methods outperforms BiLSTM model. Recent boosts of classification models revealed multiple directions, from improving the decoding modules to novel pretraining methods: (Hofmann et al., 2020) revealed the importance of learning properties of events as latent variables (for instance that the uncertainty and the mental or physical effort associated with the encounter of a snake leads to fear). Most of the recent work in emotion recognition leverage the transformer architecture, attention mechanism, convolutions and recurrent neural networks to encode different levels of information in the document. (Adoma et al., 2020) finetunes BERT on the emotion task. (Cheng et al., 2020) uses CNN to encode local features between words, and BiLSTM to encode contextual information in the sentence. (Buechel et al., 2020) presented EMOCODER, a modular decoder architecture that generalizes emotion analysis over different tasks (sentence-level, word-level, label-to-label mapping), domains (natural languages and their registers), and label formats (e.g., polarity classes, basic emotions, and affective dimensions).

3 Method

Our goal is to improve the performance and robustness of models to effectively predict fine-grained labels for emotion (Demszky et al., 2020). The following three models are designed for improvement from three directions: joint training with auxiliary tasks, redesigning architecture to explicitly attend label embedding with context, and external knowledge injection.

3.1 Baseline Model

The GoEmotions (Demszky et al., 2020) paper proposed a simple BERT based baseline model which finetunes a BERT model with 12 layers and 12 heads for classification. It feeds the output of [CLS] from BERT into a feed forward layer, and applies Binary Cross Entropy loss.

3.2 Adding auxiliary tasks

The high-level idea of this approach is to incorporate a highly-related auxiliary task to assist the learning of the main task. We introduce how to pool the losses from the two tasks (§3.2.1) and how to find auxiliary task (§3.2.2) and evaluate task similarity (§3.2.3).

3.2.1 Joint training with auxiliary tasks

Incorporating auxiliary tasks with our main task during training allows our model to leverage additional supervision and learn robust representation. Motivated by the success of multi-task learning (MTL) (Caruana, 1996) where multiple tasks are learned jointly and transfer learning (TL) (Howard and Ruder, 2018) where a model is pre-trained on an auxiliary dataset to increase the main task performance, we employ a dynamic jointly training, where the model initially has a large penalty for the loss from the auxiliary task, and then gradually emphasize the penalty for the loss from the main task. Specifically, we penalize our model with a linear combination of the losses from the auxiliary task and the main task, shown in equation 1 where L_{main} and $L_{auxiliary}$ are the losses of the main task and the auxiliary task, and the temperature T that gradually increases during training.

$$L = T * L_{main} + (1 - T) * L_{auxiliary}; \alpha \leq T \leq 1 \quad (1)$$

3.2.2 Selection of auxiliary tasks

Auxiliary task could have various relationships to the main task (Ruder, 2017). In theory, tasks are similar if the same features are used for making predictions (Caruana, 1996); or if they have the same inductive bias (Baxter, 2000); or if they have similar probability distribution and perform well in a MTL setting(Ben-David and Borbely, 2003). In practice, different hints regarding task similarity are only applicable to a specific scenario.

Figure 9 shows the correlation between ratings for each emotion and the dendrogram represents a hierarchical clustering of fine-grained emotion labels. We observed that: (1) the sentiment labeling, or coarse emotion labels, was done a priori. (2) The fine-grained label clusters closely map onto sentiment groups, showing that the two tasks could be highly related. Based on those observations, we select the classification of sentiment labels as an auxiliary task. Since the sentiment labels include fewer categories and the labels of the main tasks closely map onto sentiment groups, it is intuitive for expect the auxiliary is easier than the main task.

3.2.3 Evaluation of auxiliary task similarities

(Bjerva, 2017) and (Schröder and Biemann, 2020) estimates the effect of an auxiliary task in MTL with information-theoretic mea- sures. Both of those methods work on sequence tagging tasks. In this section, we expand those existing auxiliary task similarity measures to work for our task.

Information-theoretic measures The entropy $H(X)$ of a discrete random X in space $\mathcal{X}$ measures its unpredictability, which is defined as

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log_2 p(x). \tag{2}$$

Following (Bjerva, 2017), we consider the joint probabilities of the main and auxiliary tasks using conditional entropy. Formally, the joint probability of a pair of discrete random variables (X, Y) is defined as

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2 p(x,y) \tag{3}$$

Mutual information (MI) $I(X;Y)$ describes the ammount of information that the random variable X contains about Y, defined as

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_2 \frac{p(x,y)}{p(x)p(y)} \tag{4}$$

ith probability mass functions $p(x), p(y)$ and a joint probability mass function $p(x, y)$ (Edwards, 2008).

Clustering-based task similarity measure We measure the similarity of two tasks by calculating the comparing clusters from partitioning the validation data. According to (Nguyen et al., 2010), the measure that determine the quality of a cluster again another should quantify the amount of information shared between both clusters. Mutual information measure information shared between two clusters C and C' is derived as

$$NMI_{joint} = \frac{I(C;C')}{H(C,C')} \tag{5}$$

$$NMI_{\max} = \frac{I(C;C')}{\max(H(C),H(C'))} \tag{6}$$

(Nguyen et al., 2010) showed that NMI_{max} and NMI_{joint} satisfy the highest number of theoretical properties desirable among the clustering comparison measures. They prove that only the unit complements of both measures satisfy the metric property like positive definiteness, symmetry and triangle inequality. We used NMI_{joint} as our mutual information measure. Similar to the method of (Schröder and Biemann, 2020), we compare clusters using a contingency table.

	l'_1	l'_2	...	l'_M	Σ
l_1	c_{11}	c_{12}	...	c_{1M}	$c_{1.}$
l_2	c_{21}	c_{22}	...	c_{2M}	$c_{2.}$
...	...	...		...	...
l_N	c_{N1}	c_{N2}	...	c_{NM}	$c_{N.}$
Σ	$c_{.1}$	$c_{.2}$	...	$c_{.M}$	c

Table 1: Contingency table for a comparison of tasks L and L with N and M unique labels

Given a dataset D which is used in two tasks T and T with arbitrary label sets L and L. The comparison of L with L on D can be transformed into a clustering comparison problem. The clusters for T have N different labels, namely $l_1, l_2, ..., l_N$. The clusters for T are M different labels: $l1, l2, ..., lM$. Then, their contingency table could be represented as table 1. Different from (Schröder and Biemann, 2020), the values c_{xy} in our case are the number of data points with label l_x in task T and with label l'_y in task T'. The normalized mutual information could then be calculated as

$$\begin{aligned} NMI(L,L')_{joint} &= \frac{I(L;L')}{H(L,L')} \\ &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{c_{ij}}{c}\log_2\left(\frac{c_{ij}c}{c_{i.}c_{.j}}\right)}{-\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{c_{ij}}{c}\log_2\left(\frac{c_{ij}}{c}\right)} \end{aligned} \tag{7}$$

We use the parallel annotation from the origin GoEmotion dataset (Demszky et al., 2020) to compute the task similarity between the sentiment classification task and the goemotion emotion classification task. As a ablation study, we calculate the task similarity between emotion classification of the ekman dataset and goemotion dataset.

3.3 Attend with Label Embeddings

Let $\{E_i\}_{i=1}^{m}$ denotes the set of emotion labels in the dataset. Let D denotes a document in the dataset

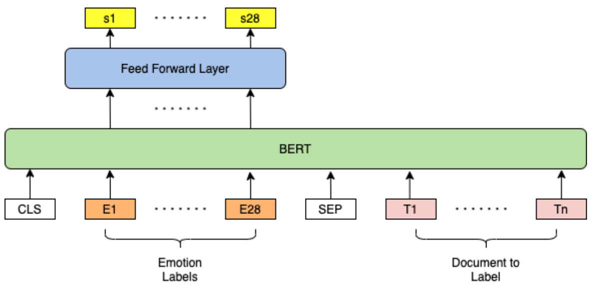

Figure 1: Model Diagram: attend input sentence with label embeddings.

that needs to be labeled with one or more emotion labels, and T_i denotes the i^{th} token in D. As shown in figure 1, we input [CLS] + [$E_1 ... E_m$] + [SEP] + [$T_1 ... T_n$] into the BERT model, and feed the output of the emotion labels into a feed forward layer to compute a score s_i for each emotion label. Different from the baseline method, we add the embeddings of the emotion labels into input, and let the attention mechanism inside BERT attend these labels with tokens in the input document. To compute the loss, we apply Sigmoid to $s_i ... s_m$, and then compute the Binary Cross Entropy loss.

This method is inspired by (Alhuzali and Ananiadou, 2021). By adding the label embeddings, we leverage the rich semantic information in the pretrained embeddings. The self-attention mechanism in BERT attends these label embeddings with each token in the document, so that different labels can focus on different parts of the document.

3.4 Embedding Mixing with Controlled Noise

This newly introduced method is inspired from multiple recent prevalent research directions including adding controlled noise and utilizing external data sources especially training sources.

3.4.1 Background

Recent approaches in using paraphrasing as a controlled noise input in improving NLP tasks performance have brought awareness of the community. Unsupervised Data Augmentation (UDA) with back-translation (Xie et al., 2019) and T5 paraphrasing (Raffel et al., 2019) are two examples of convenient tools used to paraphrase sentences. By introducing controlled noise into the input, these methods are proved to improve performance in various NLP tasks including QA and language modeling.

Inspired by the work KNN-LM proposed by (Khandelwal et al., 2020), where K nearest training sentences in the embedding space are utilized at inference time only to improve the performance of neural language models, we adopt a similar approach to taking into account the power of training knowledge when making inferences for the emotion detection task.

3.4.2 Our Mixing Approach

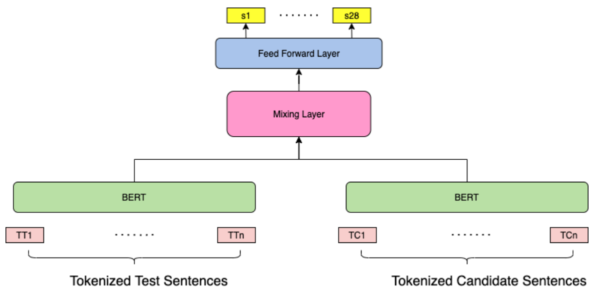

Figure 2: Model Diagram: Our Mixing Approach

Our approach combines the inspirations from the two aforementioned sources and can be visualized in Figure 2. For a given test sentence, the candidate sentence can either be computed as the top 1 similar training sentence with respect to this test sentence, or the T5 paraphrased output of this test sentence. Then both batch of sentences are passed through the same BERT encoding layers separately to get their corresponding embeddings. After that, the embeddings are added directly using a mixing weight and then pass to downstream layers. When it comes to the mixing weight, it is a fraction number between 0 and 1 and the output of the mixing layer can be simply represented in the following format: mixing weight multiplies test embeddings plus 1 - mixing weight multiplies candidate embeddings. Mixing weight is a very important hyperparameter to tune for this approach as we will discuss in more details in the results section.

We attach great importance to the validation of this method: to make sure the mixing actually works, we first set the mixing weight to be 1.0 to see if we can reproduce the results of the non-mixing version. After confirming with that, we set mixing weight to 0.5 and set candidate sentences to be the test sentences themselves and see if we can still reproduce the original results. The second validation makes sense since it is essentially mixing the test embeddings with themselves and it is expected to generate the same results as the original model. After carefully validation, we move on to incorporating T5 paraphraser and similarity metrics.

For T5 paraphraser, we currently use all of the default parameters in the original T5 paper including max length, number of returned sentences, and top k values. To find the most similar training sentence with respect to a test sentence, we use the BertScore similarity metric to compute a large matrix / dictionary which maps each test sentence to the most similar training sentence.

3.5 Method Integration

To search for the best framework, we combine the three methods presented above.

3.5.1 Label Embedding + Auxiliary Tasks

In this combination, we compute L_{main} in the same way as the Binary Cross Entropy loss in section 3.3. To compute $L_{auxiliary}$, we apply a different feed forward layer to the output of the emotion labels from BERT, to get a score for each label in the auxiliary task. Binary Cross Entropy loss is used for $L_{auxiliary}$.

3.5.2 Auxiliary Tasks + Embedding Mixing

It is very straight forward to integrate the Auxiliary Tasks Model with the Embedding Mixing Model since the only difference between these two is the loss function, which is after the definition of the mixing layer and the output layer. In other words, the model weights trained using the Auxiliary Tasks Model can be directly loaded in the Embedding Mixing Model. Also, as discussed earlier, our Embedding Mixing approach is purely inference based. Therefore, for this integrated model, we can simply change the path of the pretrained model weights and run downstream evaluation code of the Embedding Mixing approach to generate final results.

3.5.3 Label Embedding + Embedding Mixing

The key difference between these two models is that whether we want to prepend labels in front of the test sentences and candidate sentences before the tokenization steps. Therefore, we adapt the data loader class from the Label Embedding approach to the Embedding Mixing code and run downstream evaluation code of the Embedding Mixing approach to generate final results. Again, an easy way to validate this integration method is to turn off the embedding mixing by setting the mixing weight to zero and see if we can reproduce the results of a single Label Embedding model.

4 Experiments

For the experiments section, as the readers might notice, we have a rather huge configuration space with different options on a large set of possible hyperparameters. Given the time and GPU resource limit, we need to carefully select which subsets of experiments to run. Our general experimental philosophy is: start with the experiments of the three individual models first; if there is an improvement, continue to the integration stage; if not, we just stop at the current stage. We also notice that the baseline paper only ran their model for four epochs and even training this model for ten epochs can improve test performance without overfitting. To have a fair comparison between our methods and the baseline method, we include results for both epoch 4 and epoch 10 for the individual models. For the integrated models, we only include epoch 10 results which are generally better. In this section we will only focus on the important results and general trend as indicated in Table 2 and leave the detailed analysis part for next section.

4.1 Dataset

Our experiments are conducted on GoEmotions dataset (Demszky et al., 2020). There are 28 emotion labels in this dataset. The train set, dev set and test set have 43410, 5426, 5427 documents respectively.

4.2 Auxiliary Tasks

We integrated the training of the baseline model with an auxiliary module through hard parameter sharing (Ruder, 2017). The backbone model was updated according to gradients from both tasks. The temperature was initially set to be 0.8. We trained 10 epochs with a batch size of 16 and a learning rate of $5e-5$ on a 1080Ti GPU. As is in section 4.3, the threshold used is 0.3. We conducted two groups of experiment: one with the sentiment classification data as the auxiliary task (Aux Task (senti)), and the other with the ekman emotion data (Aux Task (ekman)).

4.3 Attend with Label Embeddings

For each input document, we input 50 tokens (not including emotion labels) into BERT. If the document had more than 50 tokens, only the first 50 were used. If the document had less than 50 tokens, the rest of the tokens were padded with the [PAD] token. We trained 10 epochs with batch size 16 and

Method	Epochs	Macro F1
Baseline	4	0.470
Baseline	10	0.506
Aux Task (ekman)	10	0.510
Aux Task (senti)	10	0.513
Label Embeds	4	**0.537**
Label Embeds	10	**0.537**
Label Embeds + Aux Tasks (senti)	10	0.530
Embed Mix (Train)	4	0.472
Embed Mix (Train)	10	0.472
Embed Mix (T5)	4	0.489
Embed Mix (T5)	10	0.513
Embed Mix (T5) + Aux Tasks	10	0.489
Embed Mix (T5) + Label Embeds	10	0.528

Table 2: Macro F1

learning rate 5e-5 (AdamW Optimizer) on Google Colab Tesla P100. Since our task is multi-label classification, the threshold we use to determine the labels was 0.3.

4.4 Embedding Mixing

Besides the same settings for common hyperparameters mentioned in the last two subsections, embedding mixing experiments introduce one new hyperparameters to tune: mixing weight. For both top 1 training candidates and the T5 paraphrased candidates, we experimented with a uniform distribution of mixing weight from 0.1 to 0.9 and found that a mixing weight less than 0.85 will result in a significant performance drop compared to the baseline and a mixing weight of 0.92 can reach the best performance. The mixing weight tuning process was performed for both T5 paraphraser and training similarity metric methods. It is noticeable that adopting training knowledge did not present any tendency of improvement and thus we did not proceed with it for integrated test.

4.5 Integrated Models

As stated at the start of this section, we only proceeded to integrated models once a certain model presented a tendency of improvement. As indicated in Table 2, fortunately all three models did present a certain level of improvement and there will be three possible model integrations to perform experiments on.

5 Results

In the results analysis section, we will deliver our analysis by answering the following questions: for the models with significant improvement (Label Embedding), what are the possible reasons; for the models with moderate or no improvement, what might go wrong and what could be further improved.

	Train	Val	Test
Aux (senti)	3.44e-17	0.00460	0.00010
Aux (ekman)	1.78e-05	0.00833	0.00028

Table 3: Normalized Mutual Information between auxiliary tasks and the main task.

5.1 Adding Auxiliary Tasks

As shown in table 3, the normalized mutual information of both tasks are positive, showing that the tasks are similar to each other. Also, both pairs of tasks have close values in the validation set and the test set. As shown in table 2, both Aux Task (ekman) and Aux Task (senti) lead to improvements to the baseline model. However, we did not observe a positive correlation between the similarities of tasks and the performance improvements: the ekman task has a higher task similarity but a lower performance increase. It is possible that we are working on a multi-label classification task: say we have two data points: one with labels 1, 2 and the other with labels 1. The 1 in the first point should be assigned a higher weight, but when we calculate the NMI, we assign the same weight to the three labels. Further improvement to the definition of NMI is needed to normalize this effect.

5.2 Label Embedding

As shown in table 2, label embedding method gives the best macro f1. The 4 epoch version beats our baseline macro f1 by more than 6%. The 10 epoch version beats the baseline 10 epoch version by 3%. To understand where the improvements come from, we compare the f1 score of each class given by this new model against the f1 given by the baseline model in figure 3. We observe significant differences in the f1 scores of class 16 (grief), 21 (pride), 23 (relief). As shown in figure 4, these three classes are the ones with fewest examples in the train dataset. Our method is able to improve the performance on the low frequency classes. We think the model is able to exploit the semantic information in the pre-trained embedding of emotion labels, so that it does not require as many training instances in the fine-tuning stage.

Our model also converges faster than the baseline. In figure 5, we plot the macro f1s by the training steps. Our model (label_emb) reaches the best f1 at step 8000 (within 4 epochs), while the baseline reaches its best f1 at step 26000. Our model takes one-third of the time to reach a better performance than the baseline. Moreover, we observe that the f1 of our model is much higher than the baseline at the beginning of training. This confirms our hypothesis that the model can benefit from the information in the embedding.

To interpret our model, we visualize the attentions in the model. We chose one document that is correctly labeled by our model, but incorrectly labeled by the baseline. Figure 7 shows the first three and last three attention layers in our model. In the first several layers, the attention flows into [CLS] or tokens in the input document. In the last several layers, the attention flows to the predicted emotion label. The document we visualize is labeled as fear by our model, and labeled as neutral by the baseline. Its ground truth label is fear. In figure 6, we observe that the attention flows from the word "fear" in the document to the emotion label fear. However, while the baseline model uses the output from [CLS] for classification, the attention in the last layer that flows into [CLS] is weak (figure 8). We think the information that ends up in the [CLS] token in the baseline model is not as useful.

5.3 Embedding Mixing

As shown in Table 2, there is no significant improvement after we mix most similar training embeddings with each of the test embedding. We attribute this behavior to the following possible reasons: first, the domain transfer from language modeling might not work properly with classification models; second, in the KNN-LM and related papers (Khandelwal et al., 2020), the authors also attempt using top-n training examples instead of just top-1; third, it is possible that other similarity metrics such as ROUGE scores and even syntactic similarity measures could work better in this case. We have already set up the code for mixing with multiple candidates but it takes much time to inference due to the time limit. It is also worth while to try different similarity metrics and ensemble their scores for selecting the final candidates in the future.

In sharp contrast to the mediocre performance of training example based mixing, T5 mixing actually presents an non-trivial improvement. As stated in the methods section, paraphrasing is essentially adding controlled noise to the model to help it generalize better. Given the current results, it would be optimistic to try either mixing with multiple T5 candidates (since one test sentence can generate multiple paraphrases) or UDA candidates.

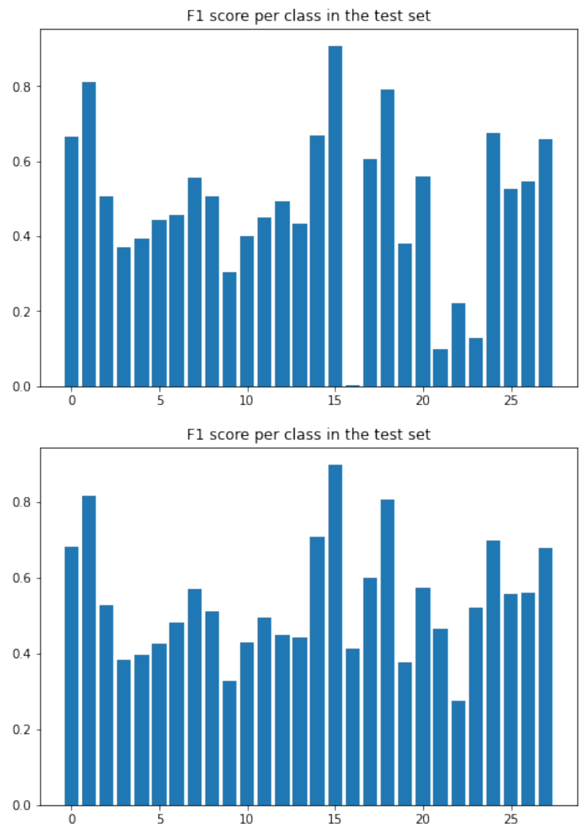

Figure 3: F1 Score for Each Class. Top: Baseline Model (4 epochs). Bottom: Label Embedding Method (4 epochs).

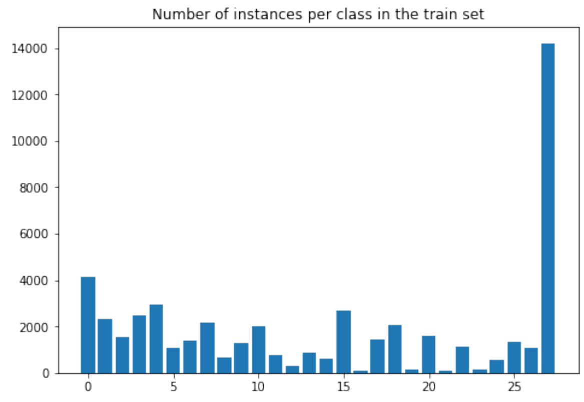

Figure 4: Number of Train Instances for Each Class

5.4 Integrated Models

Although from Table 2 we can see that after pairwise integration, there is usually an improvement

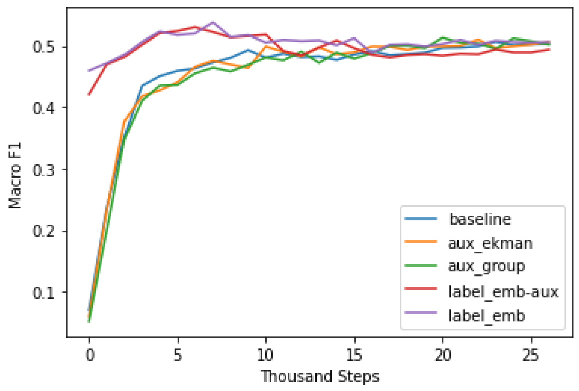

Figure 5: Macro F1 for every 1000 training steps on the testset.

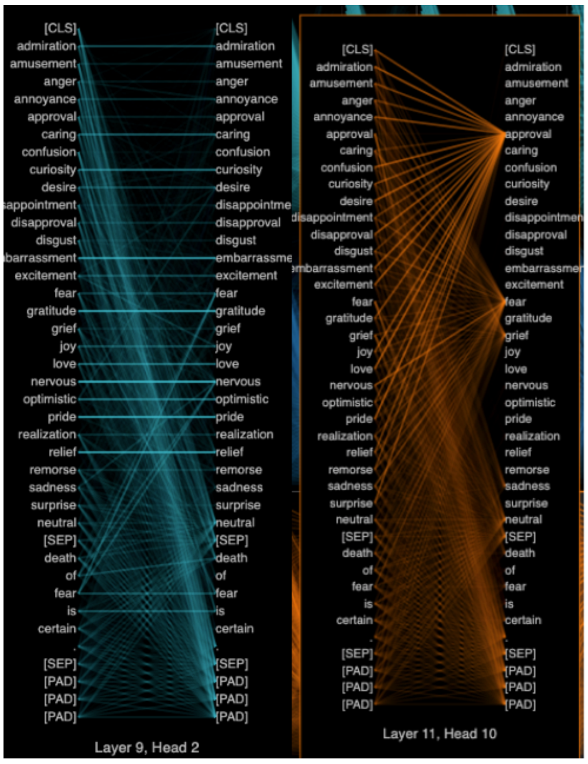

Figure 6: Attention of the Label Embedding model. Ground truth label and the predicted label of the sentence are both ”fear”. Left: Layer 9, Head 2. Right: Layer 11, Head 10. Layer and head indices start from 0.

compared to Auxiliary Tasks or Embedding Mixing alone, we still cannot beat the scores for just using Label Embeddings. During the integration of T5 mixing with Auxiliary Tasks, we even see a drop in performance. Therefore, it could be concluded that the integration of the three models of completely different directions might not work in this case.

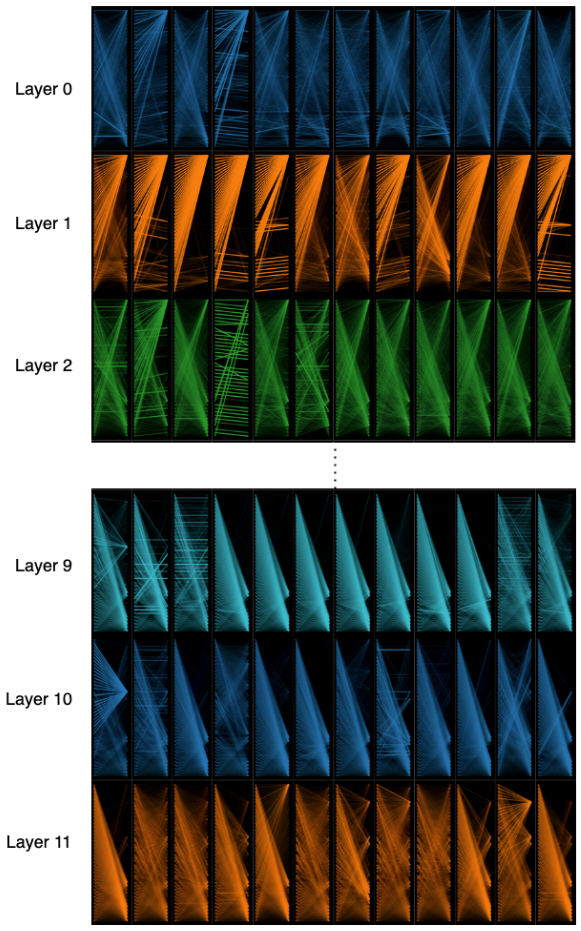

Figure 7: First three and last three attention layers of the Label Embedding model. On each layer, the plot shows head 0 to head 11 from left to right.

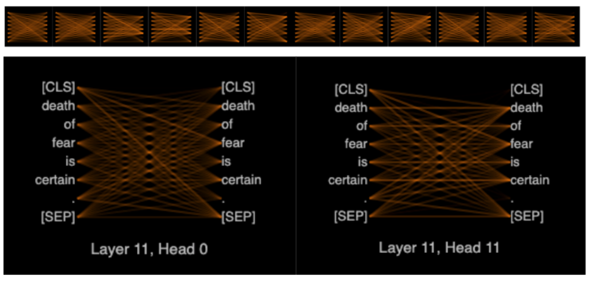

Figure 8: Attention of the baseline model. Top: layer 11. Bottom Left: Layer 11, Head 0. Bottom Right: Layer 11, Head 11.

6 Conclusion

In this paper, we examined the classification problem of 28 emotions for the most recent benchmark GoEmotions (Demszky et al., 2020) dataset. By proposing and implementing three modeling assumptions of completely different directions, we found that there is a significant amount of space for improvement for this dataset and our best performing method can achieve a 6% increase in average macro-f1 score. We have also listed a few directions for future improvements of the three methods in the result analysis section. Even though we have promising initial results, we still believe that more complete per case based statistical significance tests and more experimental iterations with different seeds should be carried out before we move to more complex modeling options and experimentation.

7 Appendices

Label Name	Label Index
admiration	0
amusement	1
anger	2
annoyance	3
approval	4
caring	5
confusion	6
curiosity	7
desire	8
disappointment	9
disapproval	10
disgust	11
embarrassment	12
excitement	13
fear	14
gratitude	15
grief	16
joy	17
love	18
nervousness	19
optimism	20
pride	21
realization	22
relief	23
remorse	24
sadness	25
surprise	26
neutral	27

Table 4: Mapping between Label Name to Label Index

References

Acheampong Francisca Adoma, Nunoo-Mensah Henry, Wenyu Chen, and Niyongabo Rubungo Andre. 2020. Recognizing emotions from texts using a bert-based approach. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 62–66. IEEE.

Hassan Alhuzali and Sophia Ananiadou. 2021. Spanemo: Casting multi-label emotion classification as span-prediction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1573–1584.

Jonathan Baxter. 2000. A model of inductive bias learning. *J. Artif. Intell. Res.*, 12:149–198.

Shai Ben-David and Reba Schuller Borbely. 2003. Exploiting task relatedness for mulitple task learning. In *COLT*.

Johannes Bjerva. 2017. Will my auxiliary tagging task help? estimating auxiliary tasks effectivity in multi-task learning. In *NODALIDA*.

Laura Ana Maria Bostan and Roman Klinger. 2018. An analysis of annotated corpora for emotion classification in text. In *COLING*.

M. Bradley and P. Lang. 1999. Affective norms for english words (anew): Instruction manual and affective ratings.

Sven Buechel, Luise Modersohn, and U. Hahn. 2020. Towards a unified framework for emotion analysis. *ArXiv*, abs/2012.00190.

R. Caruana. 1996. A dozen tricks with multitask learning. In *Neural Networks: Tricks of the Trade*.

Yan Cheng, Leibo Yao, Guoxiong Xiang, Guanghe Zhang, Tianwei Tang, and Linhui Zhong. 2020. Text sentiment orientation analysis based on multi-channel cnn and bidirectional gru with attention mechanism. *IEEE Access*, 8:134964–134975.

Dorottya Demszky, Dana Movshovitz-Attias, J. Ko, Alan S. Cowen, Gaurav Nemade, and S. Ravi. 2020. Goemotions: A dataset of fine-grained emotions. In *ACL*.

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

Sheri Edwards. 2008. Thomas m. cover and joy a. thomas, elements of information theory (2nd ed.), john wiley sons, inc. (2006). *Inf. Process. Manag.*, 44:400–401.

P. Ekman. 1992. An argument for basic emotions. *Cognition Emotion*, 6:169–200.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Jana Hofmann, Enrica Troiano, K. Sassenberg, and Roman Klinger. 2020. Appraisal theories for emotion classification in text. In *COLING*.

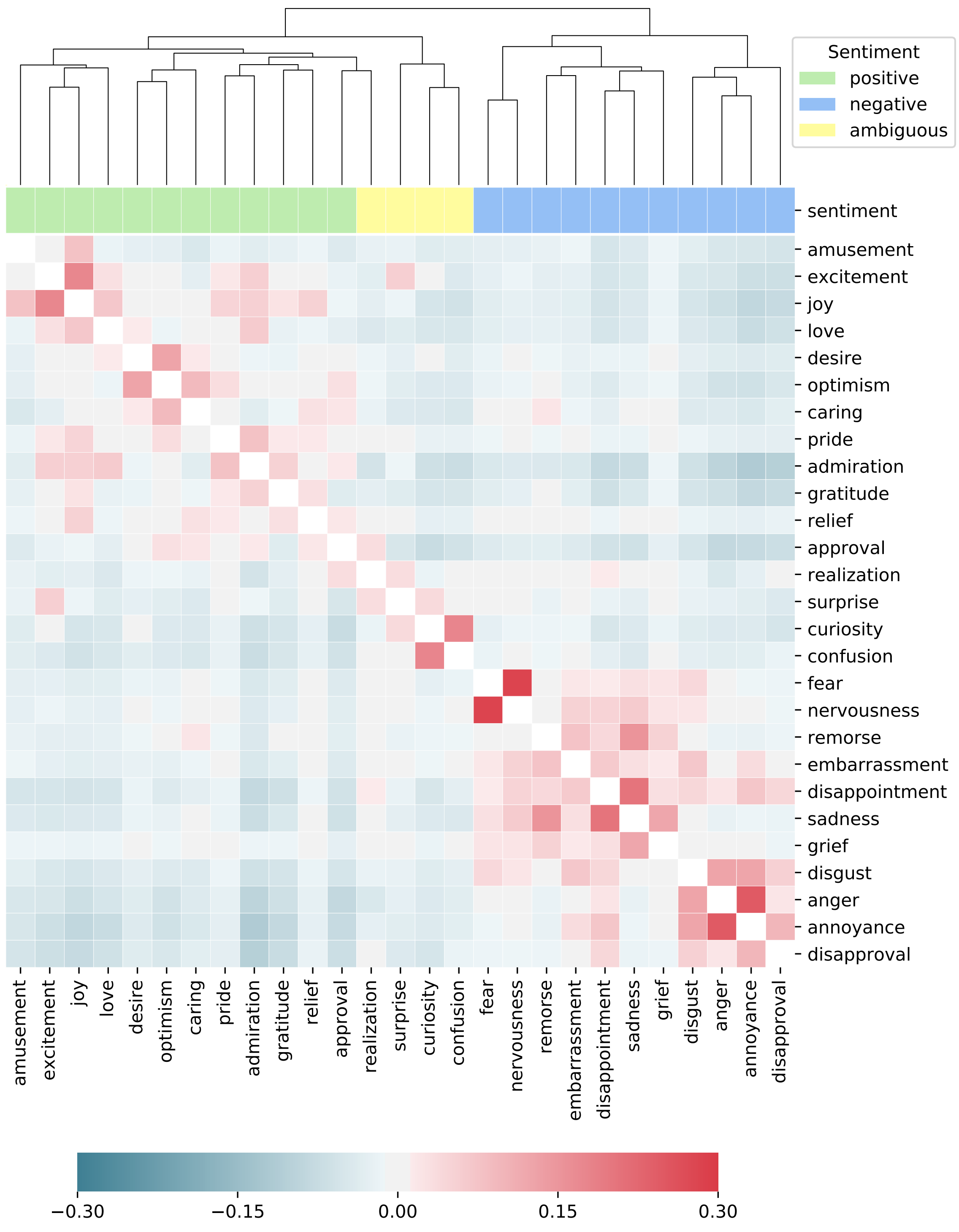

Figure 9: The heatmap shows the correlation between ratings for each emotion. The dendrogram represents the a hierarchical clustering of the ratings. The fine-grained label clusters closely map onto sentiment groups, showing that the two tasks could be highly related.

J. Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models.

Yuan Lin, J. Li, L. Yang, K. Xu, and Hongfei Lin. 2020. Sentiment analysis with comparison enhanced deep neural network. *IEEE Access*, 8:78378–78384.

Saif M. Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20, 000 english words. In *ACL*.

X. V. Nguyen, J. Epps, and J. Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854.

R. Plutchik. 1984. Emotions : a general psychoevolutionary theory.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *ArXiv*, abs/1706.05098.

Fynn Schröder and Christian Biemann. 2020. Estimating the influence of auxiliary tasks for multi-task learning of sequence tagging tasks. In *ACL*.

C. Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *SemEval@ACL*.

W. Wang, L. Chen, K. Thirunarayan, and A. Sheth. 2012. Harnessing twitter ”big data” for automatic emotion identification. *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 587–592.

Amy Beth Warriner, V. Kuperman, and M. Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior Research Methods*, 45:1191–1207.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.