From Words to Feelings, Emotion Prediction using Classical and Deep Learning Models

Sohaib Bantan
Department of Computer Science and Eng
, University of Minnesota, Twin Cities
banta033@umn.edu

Azzam Alharthi
Department of Computer Science and Eng
, University of Minnesota, Twin Cities
alhar112@umn.edu

I. QUESTIONS

- Is the task of detecting emotion from short form text feasible using machine learning and deep learning techniques, and how reliable are these performances when mapping the twenty six GoEmotions labels into Dr Paul Ekman's six emotion groups?
- How well does a PyTorch based neural network compare against a classical machine learning model in identifying fine grained emotions in short text samples?
- How well can these models generalize to longer and more expressive text samples?

II. DATASET

GoEmotions is a large-scale emotion classification dataset manually developed by Google Research [1]. The dataset contains 58,009 English-language Reddit comments collected from a wide range of subreddits between 2005 and 2019. Each comment is annotated manually by 3-5 trained raters. The annotations were done on 27 emotion categories, and an extra one as neutral. As mentioned, Reddit was selected as the source of this dataset. We believe that this is suitable since it contains a high volume of conversational content, where emotions are naturally present.
Google performed data-curation measures to ensure the dataset does not reinforce general, nor emotion-specific, language biases. This is because the known demographic of Reddit users is mostly young male users, which does not make it a globally diverse population. To address this, Google identified harmful/offensive language using predefined terms, filtered them out and masked them. They additionally filtered the data to reduce profanity, limit text length, and balance for represented emotions and sentiments.
The dataset consisted of 43,410 training samples and 5,427 testing samples. Each comment/ text had a max length of 30 words. The structure of one sample was in the form of [Comment in text form, ID representing the respective emotion class (0-27), a unique identifier]. These samples were placed in a CSV file, where each line corresponded to a specific sample.
For data preprocessing, we mapped the 28 emotions into Dr. Paul Ekman's 7 emotional groups: joy, anger, sadness, disgust, surprise, fear, neutral. We then loaded the dataset from HuggingFace's datasets library, split it into three data frames, and added two columns for the new ekman labeling. Moreover, we did the following for each model:

LinearSVM:

- No manual preprocessing, all done by TF-IDF, which is a vectorizer that converts text into vectors based on importance.
- This includes tokenization, normalization, lowercasing, and n-gram expansion.

CNN:

- The text sequences were first tokenized, whilst cleaned by keeping only basic punctuation and expanding words like you're to you are. We then build a vocabulary from the training data by assigning each word to a unique integer index.
- We then encode each sentence by turning it into a fixed-size list of numbers, shortening it if it is too long and padding it if it is too short so the CNN can process it.
- We finally load a pre-trained embedding layer called GloVe, to create the embedding matrix for the encoded tokens, which essentially allows for the representation of the different tokens.
- For the batch size, we set it to 32 for the CNN implementation.

BERT:

- We create a dataset class that prepares the data for the BERT-based model by tokenizing each text and shortening or padding it to a fixed size of 64 tokens.
- The class also returns input IDs, attention masks which identify real words from padding, and the corresponding emotion label.

III. Background and Related Work

We studied the paper Linear-layer Extrapolation for Fine-Grained Emotion Classification by Maukh Sharma et al [2]. Their motivation was to improve transformer models on detecting fine-grained emotions. Based on previous work, they identified that the later layers of a neural network made more successful predictions than the early layers. They used a contrastive classification function, one that used the expert layer predictions (later) and the amateur layer predictions (earlier) and used a specific contrastive strength parameter.
The layer whose prediction distribution diverges the most from the final layer is automatically selected as the amateur layer. Then, they used multiple datasets, including GoEmotion, to fine-tune pre-trained transformer models: this included DeBERTa-XL and FLAN-T5-XL. For metrics, they reported the F1, recall, and precision of each label, focusing on the rare classes. Their findings matched their theories, where using contrastive classification improved the recall and macro F1. However, they were limited to moderate-sized models, with potential future work focusing more on larger models.

IV. Models

A. *LinearSVM*

The reason we chose LinearSVM is its strong performance on text classification. The dataset used contains a lot of short text samples, where LinearSVM can compete with even complex neural networks. It is also not very computationally demanding, which is important for our limited resources setup.

B. *Convolutional Neural Network (CNN)*

We chose to implement a CNN because we were curious to see how a PyTorch based convolutional neural network would perform on the task of text based emotion detection, rather than its more common application of visual tasks where it is known to perform well. Shown in the training approach section, Fig 1 summarizes the design of the CNN model. Nevertheless, CNNs are also widely used in natural language processing tasks due to their ability to capture patterns in data, making them a natural choice for this problem.

C. *Bert-based model*

Several past studies for our task finetuned pretrained BERT models on multiple datasets, including GoEmotions [1] [3]. They found that this approach yielded the state of the art results. Based on these results, we implemented a BERT-based model as a well-established baseline for comparisons.

V. Training Approach

A. *LinearSVM*

We used a combine TF-IDF feature extractor that integrates word and character level representations. The word and character level vecotrizers relied on word and character n-grams respectively. They were then merged using FeatureUnion.
Then the labels were converted to binarized format using the MultiLabelBinarizer. This enables the OneVsRest classifier, which was required by LinearSVC. Lastly, class balancing was chosen while training the classifier.
We used GridSearchCV over a pre-chosen range of hyperparameters to determine the best choices for the model. It evaluated the best model based on the macro F1 score in the validation split. These were the hyperparameters and their chosen ranges:

- Max Features (word): 20,000 40,000
- n-gram Range (word): (1,2) (1,3)
- Max Features (char): 10,000 20,000
- n-gram Range (char): (3,4) (3,5)
- C Estimator: (0.1, 0.3)

B. *CNN*

Using PyTorch to design the model, we began with an embedding layer of 100 dimensions imported from GLoVe. Three 1D convolutional layers with kernel sizes of 3,4 and 5 were applied, utilizing 128 filters. Following the convolutions were ReLU activation functions to introduce nonlinearity. Global max pooling was then used to activate the strongest feature map produced by the kernels. The outputs of the three layers were concatenated and passed through a dropout layer with a rate of 30%. This was done to reduce overfitting. Finally, a fully connected layer was used to output the final emotions class probabilities. This is where the labels were utilized. Figure 1 illustrates the design of our CNN model and summarizes the above.

To finetune the hyperparamters, we implemented a manual grid search by testing different embedding sizes, number of filters, dropour rates, and learning rates. Each case was trained for up to 5 epochs with early stopping. Taking into account the hyperparamter tuning, the following are the hyperparameters used:

- 100 dimension GLoVe embedding layer
- Three one dimensional convolutional layers with kernel sizes 3, 4, and 5
- 128 convolutional filters per kernel size
- Dropout rate of 0.3
- Learning rate of 0.0005
- 10 maximum epochs

This model was trained using the Adam optimizer, in addition to cross-entropy as our loss function. To prevent overfitting, we implemented early stopping such that if the validation loss did not improve for two consecutive epochs, it would stop and store the best model state. As for the class imbalance, where we had much more data points for the joy, neutral, anger, and surprise classes, we utilized class weights, however we found that the results were worse. Due to that, we did not implement any class imbalance solutions.

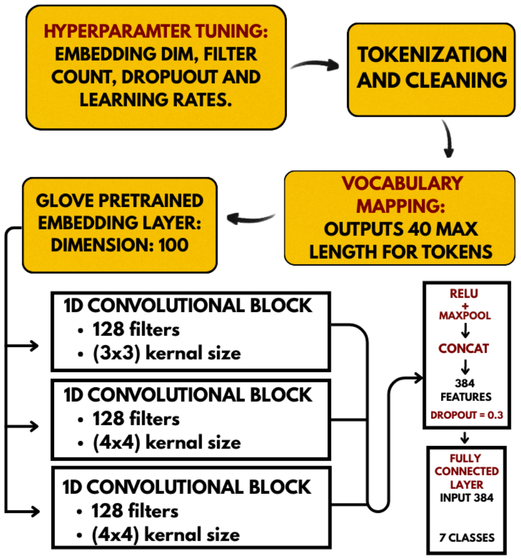


Fig. 1: CNN architecture for emotion classification.

C. BERT-based

We used a pretrained bert base case model, such that it was orignally trained on large scale English text using supervised tasks. We tokenized each text, applying truncation and padding to a fixed size of 64 tokens, along with attention masks that identify real tokens from padding. The model was setup such that it classifies at the end, which is what produces our desired output and to where the fine-tuning to our task really happens. It was trained for up to ten epochs using the AdamW optimizer with a learning rate of 2e 5 and cross entropy loss. Similarly to the CNN approach, we also used early stopping.

VI. Results

A. Evaluation Metrics

- Macro F1: This metric calculates the average F1 score across all labels, ignoring the label's frequency. This is important in a slightly imbalanced dataset, which is what we are working with.
- Precision: This metric is crucial for detecting rare classes, since it calculates how many false negatives were avoided.

- Recall: It is very important since it tells us how many false predictions were made for a specific label. A high recall means the model avoids false positives, which is important for under-represented classes.
- Accuracy: Even though it is not a very meaningful metric in this case, since some labels are more common than others, it still provides a simple overview of the model's performance.

You can also include a second table for Ekman based groups.

B. Per Class Analysis

Include per class F1 scores, either in a table or in a figure. Comment on which emotions are easier or harder to predict and why that may be the case.

TABLE I: LinearSVM Test Performance per Emotion Label

Emotion	Precision	Recall	F1 Score	Frequency
Anger	0.42	0.64	0.50	726
Disgust	0.43	0.54	0.48	123
Fear	0.54	0.70	0.61	98
Joy	0.81	0.76	0.78	2104
Neutral	0.57	0.78	0.66	1787
Sadness	0.44	0.61	0.51	379
Surprise	0.45	0.65	0.54	677
Macro Average	0.52	0.67	0.58	5894
Weighted Average	0.61	0.73	0.66	5894

It can be seen that joy got the highest metrics out of all the other emotions, and that is because it has the highest frequency. Neutral also has a high frequency, and it has the second highest F1 value. This means the LinearSVM model can reliably identify these emotions. On the other hand, disgust has the worst F1 score out of all the emotions. That can be directly linked to its low frequency, with only having 84 entries. However, fear also has a low support score, but still got a good F1 score of 0.61. What might be the reason? It is the distinct vocabulary associated with fear that are easy to label: like scared, terrified, and afraid. However, disgust has some shared vocabulary with other emotions: like weird and nasty.

TABLE II: CNN Performance per Emotion Label

Emotion	Precision	Recall	F1 Score	Support
Anger	0.57	0.32	0.41	703
Disgust	0.69	0.21	0.33	84
Fear	0.70	0.47	0.56	90
Joy	0.72	0.84	0.77	2054
Neutral	0.57	0.64	0.60	1606
Sadness	0.54	0.45	0.49	317
Surprise	0.56	0.48	0.52	573
Macro Average	0.62	0.49	0.53	5427
Weighted Average	0.63	0.64	0.62	5427

Similar to the results for the previous SVM model, joy got the highest metric out of the other classes with a score of 0.77 F1 score. However, we find that the performance generally dipped for all classes. An interpretation of why that happened is due to the complex structure of the CNNs, such that the bigger the data the better the performance. Generally, strong results showed up for when the frequency was high. This indicates the extent of the effect of the data imbalance. Overall, it produced a macro avg of 0.53, which shows that it is underforming compared to the SVM model.

TABLE III: BERT Performance per Emotion Label

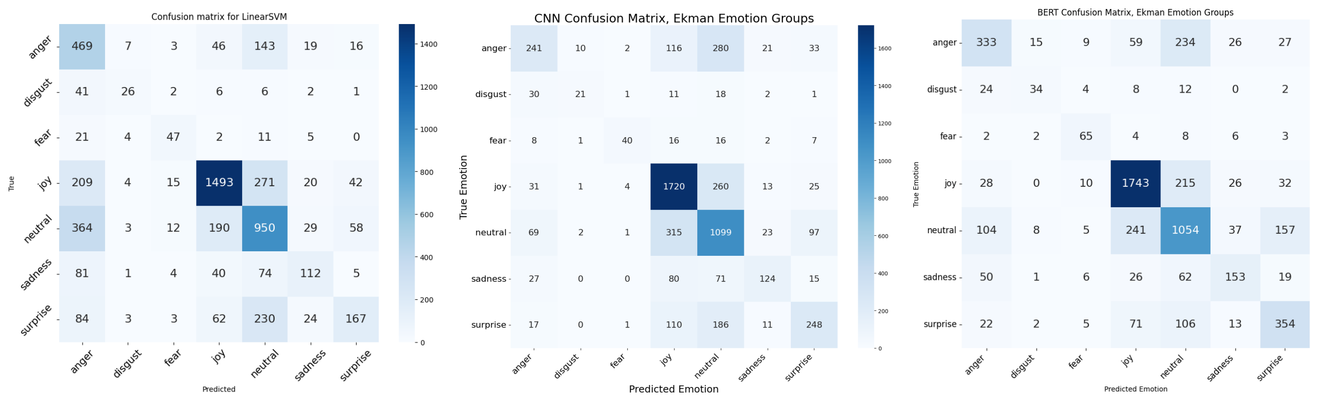
Emotion	Precision	Recall	F1 Score	Support
Anger	0.63	0.45	0.52	703
Disgust	0.45	0.42	0.43	84
Fear	0.70	0.57	0.63	90
Joy	0.82	0.83	0.82	2054
Neutral	0.60	0.72	0.65	1606
Sadness	0.59	0.54	0.56	317
Surprise	0.63	0.51	0.56	573
Macro Average	0.63	0.58	0.60	5427
Weighted Average	0.69	0.69	0.68	5427

BERT gave the best performance, with a macro F1 of 0.6. It gave the highest metrics for joy, which shows it can learn contextual cues the most out of all the models. It also gave the best score for surprise, with an F1 of 0.56. Lastly, the huge difference between the macro F1 and the weighted F1 shows how much the model would improve with a properly balanced dataset. It would reach a great macro F1 of 0.68.

It is worth mentioning that the results between the models were significant enough to be worth it. Furthermore, the results between the validation and test stages were consistent.

VII. Visualizations

A. *Confusion Matrices*



(a) LinearSVM (b) CNN (c) BERT

Fig. 2: Confusion matrices for CNN, LinearSVM, and BERT across Ekman emotion groups

VIII. Discussion

A. *Limitations*

All models struggled with emotional ambiguity between similar emotions' vocabulary. A more specific explanation for each model is listed below:

- LinearSVM struggled with sarcastic statements and ones that require context.
- CNN is usually used for image classification, which means it receives much more data than what was present in these text samples. Therefore, when receiving this lower number of data, it performs worse.
- Even though BERT gave the best results, it also had the highest runtime as well as computational costs.

B. *Use Cases*

- LinearSVM can be used in environments where computational efficiency is crucial, like analyzing customer reviews. That is because it had the fastest runtime out of all the models.
- CNN can be used with larger datasets, providing it with enough entries to learn the patterns. We predict a large enough dataset will allow CNN to outperform both models used in this project.
- BERT's good performance means it can be used in environemnts where accurate predictions are more important than fast computation. For example, mental health analysis and crisis detection.

C. *Ethical Implications*

The GoEmotions dataset is made up of Reddit comments, which are usually written by young white males. This means the models will encode cultural and linguistic biases aimed at this demographic.
The biggest risk is failure at detecting sarcasm, which our models struggled with. This can lead to false responses when applied to mental health evaluation or content moderation.
Lastly, a grave mistake is the over reliance on these models' predictions and treating them as ground truth instead of mere estimations.

References

[1] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "Goemotions: A dataset of fine-grained emotions," *arXiv preprint arXiv:2005.00547*, 2020.

[2] M. Sharma, S. O'Brien, and J. McAuley, "Linear layer extrapolation for fine-grained emotion classification," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 20 880–20 888.

[3] S. K. Bharti, S. Varadhaganapathy, R. K. Gupta, P. K. Shukla, M. Bouye, S. K. Hingaa, and A. Mahmoud, "Text-based emotion recognition using deep learning approach," *Computational Intelligence and Neuroscience*, vol. 2022, no. 1, p. 2645381, 2022.