IMPROVING SENTIMENT CLASSIFICATION USING 0-SHOT GENERATED LABELS FOR CUSTOM TRANSFORMER EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present an approach to improve sentiment classification for transformers (based on BERT and DistilBERT) using additional embeddings to represent emotion inputs. We used HuggingFace's 0-shot prediction pipeline to generate probabilities of whether emotions apply to a given sample. We generated 0-shot probabilities for 1.6 million samples from a sentiment classification dataset and a smaller sentiment airline dataset using 63 emotions. Then we added custom tokens to BERT's embeddings and tokenizers representing various levels of emotion for each predicted emotion. Finally, depending on the probability of each emotion, the respective custom token representing that level was prepended to the text input of the model to process and train for classification. We additionally test direct classification layer addition of emotion inputs and an ensemble of BERT and DistilBERT models both using emotion inputs achieving a modest increase in sentiment prediction accuracy. Our results show modest improvement in all cases over the original model for both BERT and DistilBERT tested with added emotion inputs generated from 0-shot pretrained models.

1 INTRODUCTION

Natural language processing (NLP) is a rapidly advancing field due to the developments of the Transformer model Vaswani et al. (2017) using a combination of encoder and decoder to process tokenized text with attention. Some popular models use bidirectional embeddings like BERT Devlin et al. (2018) which is composed of just the encoder, while others use just the decoder like GPT-2 Radford et al. (2019) and GPT-3 Brown et al. (2020). All of these models are composed of a large number of parameters and this number of parameters has tendency to increase with time from several hundreds of millions to more than one hundred billions (GPT3). Recently, (while not the main goal for some models) another tendency focuses on reducing parameter sizes and efficiency like with Reformer Kitaev et al. (2020), RoBERTa Liu et al. (2019) and DistilBERT Sanh et al. (2019), as well as to focus on increasing the input sequence length such as with Transformer-XL Dai et al. (2019) and XLNet Yang et al. (2019).

Transformers are the state-of-the-art in many NLP tasks. In this paper we focus one of these tasks, sentiment classification, to study how additional semantic information could improve the prediction. Sentiment classification is important for a variety of tasks. For individual samples, sentiment classification can be used to monitor comments on a platform (social media, direct messaging in sales, etc.). They can also be used to gather general sentiment from a user's history of comments, or expand to the sentiment of the public regarding topics like politics or world events from general social media scraping. Sentiment classification is generally performed by predicting a positive or negative label for sentiment, or a probability representing 1 as positive and 0 as negative. Some instances will have a neutral output incorporated as a third label represented as 1, 0 and -1 for positive, neutral, and negative, respectively.

Emotion classification aims at predicting emotion labels for text rather than a general sentiment. Emotions such as joy, anger, neutral, disgust, fear, sadness and surprised are commonly used in addition to other common emotions like love, hate, worry and relief. Often emotion classification datasets are small in size with extremely unbalanced labels, but also suffer from labeller interpre-

tation of complex emotions. Many datasets only include a small number of emotions and/or one emotion per sample, but text can have a variety of emotions, each at different levels. Emotions are highly subjective in text especially when it comes to choosing one emotion to represent one sample since human labelers can classify the text differently between one another depending on the interpretation or impression left on the person.

We aim to improve sentiment predictions using enhanced emotion labeling by incorporating them as tabular data inputs to transformers. Emotion datasets provide poor quality labeling whether through inconsistent interpretation or selection of a single emotion to represent the sample leading to sparsity of labels. By improving the emotion labeling to encompass multiple emotions with a range of emotion levels (without limitations on which emotions we can use), we can apply these labels to sentiment classification allowing the model to take a wide range of emotions into consideration which directly correlates to sentiment.

This paper is organized as follows. Section 2 describes related previous works. We describe our sentiment models' design using our dataset with generated emotion labels in Section 3. Experiments, results and comparisons are presented in Section 4. Finally, the paper is concluded in Section 5.

2 PREVIOUS WORKS

Transformers advanced NLP significantly since the original paper "Attention Is All You Need" Vaswani et al. (2017). The transformer uses attention and self-attention mechanisms in order to detect and model dependencies and similarities between input words that are used to construct a dynamic embedding of all the words. Unlike Recurrent Neural Networks (RNN), the model has a pre-set input sequence length, and is generally trained using significantly larger models. Despite these drawbacks, transformers have outperformed RNNs significantly.

Transformers are generally trained on large corpora, such as Wikipedia, containing millions of articles and billions of words. Due to the large size of the training data coupled with the large size of transformer models, it requires significant resources and time to train. Rather than training a transformer from scratch, we look to pre-trained models with proven performance on several NLP tasks.

Our results are demonstrated using the pre-trained models BERT Devlin et al. (2018) and DistilBERT Sanh et al. (2019). As BERT has been a very popular model and has performed very well since its release, we chose BERT and DistilBERT to highlight improvements in these models while showing how we can improve the smaller distilled variation using the same approach to compete with the original BERT model. The base uncased BERT model has 110 Million parameters and 340 Million for the large model while DistilBERT reduces the size of a BERT model by 40% while achieving a similar performance.

Sentiment classification is a popular task but is often a difficult task due to the way humans express themselves in style and meaning. Naseem et al. (2020) use a combination of several model embeddings to overcome this problem and train a bi-LSTM using embeddings from character LSTM, (word) GloVE, parts of speech (POS), Lexicon, (context) ELMo and (transformer) BERT. This comparison Zimbra et al. (2018) of several SOTA approaches shows the difficulty modelling sentiment with nearly all approaches performing under 70% averaged over 5 sentiment datasets. Yin et al. (2020) use a BERT model for sentiment classification by applying a two layer attention system to the outputs of the BERT encoder at the token and phrase level using a parse tree. There's a focus on sentiment classification using BERT in different languages such as mixed English and Spanish (Spanglish) Palomino & Ochoa-Luna (2020) and Arabic Antoun et al. (2020). Transformers (like BERT) and other models have also been tested for the low-resource language Bangla for sentiment classification Arid Hasan et al. (2020).

To provide additional information to our model, we use an approach to incorporate tabular data into transformers. There have already been some works involved in embedding tabular data into a transformer model. Kiela et al. (2019) use the idea of embedded tokens to add pooled outputs from a ResNet He et al. (2015) model into a transformer allowing the transformer to process both text and associated images. Similarly, Lu et al. (2019) also embeds image information by using a Fast(er) R-CNN Ren et al. (2016) to produce features, but the features are added to the embeddings of the

text. They mask regions of interest and also provide them as added features to custom "[IMG]" tokens allowing the model to predict information about these masked regions.

3 METHODOLOGY

In this section we describe our methodology in two steps. The first step is generating emotion probabilities for the sentiment dataset we intend to train on, in addition to preprocessing these emotions for use in the transformer. The second step is our training procedure to classify the sentiment dataset using these additional emotion features. Our approach uses the idea of custom embeddings to include emotions as token inputs into the transformer for sentiment classification. We designed our approach to handle a range of emotional inputs for every token using multiple embeddings within a specified emotion range, in addition to emotional inputs added to the classifier layer. We generate 63 emotion categories at random in our tests automatically generated without any training (0-shot) relieving limitations on the number of emotion categories or the emotions used. Although there's no limit to the number of emotions, aside from BERT's maximum input length of 512 which must also include text, we limited our tests to 63 emotions to include all information. We also show the change in accuracy by selecting subsets of emotions which highlight more emotions provide more accuracy.

3.1 DATASETS

We use the sentiment140 Go et al. (2009) dataset which consists of 1.6 million samples classified as 0 for negative and 4 for positive (relabeled to 1) with an even class distribution. We chose it because it's the largest tweet sentiment dataset with a general purpose topic (not specific to reviews or categories). The samples were scraped from twitter and include separate information on the user that posted the tweet such as their username and date, in addition to their tweet text. We discard user information and data information and simply use the tweet text for sentiment classification. We do not alter the text in any way like removing links, user mentions or symbols, we use the dataset as it is presented.

We additionally test on a smaller dataset US-Airline sentiment from crowdflower [1]. The dataset consists of tweets mentioning US airlines and are labeled for sentiment. There are 2363 positive samples, 9178 negative samples and 3099 neutral samples. We removed the neutral samples since our approach is built for positive or negative predictions, leaving us with 11,541 samples. Due to the imbalance in classes, we removed most of the negative samples to match the size of the positive class giving us 4726 samples.

3.2 DATA PRE-PROCESSING

We generate emotion categories for each sample in the dataset for the first step in our pre-processing. We chose a random list of 63 general emotions for our examples which include some common emotions found in emotion datasets. We didn't choose any particular set of emotions because we wanted to capture a wide range of possible informative features to the dataset, but we also try removing emotions not specific to positive or negative sentiment. Datasets for emotion classification like the dataset generated from crowdflower and from Saravia et al. Saravia et al. (2018) use just a single emotion for each sample (sometimes including an "other" category), usually from these typical emotions: *empty, sadness, enthusiasm, neutral, worried, surprise, love, fun, hate, happiness, boredom, relief, anger*. We include all of these emotions aside from "neutral" and "other" and added many more: *amusement, annoyed, anxious, aversion, bitter, cheated, compassion, confused, contentment, contrary, desperate, disappointed, disapproving, disgust, dislike, disturbed, doubtful, excitement, fear, frustrated, gloomy, grieved, heartbroken, hopeless, horrified, infuriated, insulted, irritated, joy, loathing, lonely, lost, mad, miserable, nauseated, nervous, offended, panicked, peace, peeved, pride, resigned, revulsion, satisfaction, stressed, terrified, troubled, uncomfortable, unhappy, vengeful, withdrawal*.

[1] https://www.crowdflower.com/data-for-everyone/, https://data.world/crowdflower/sentiment-analysis-in-text

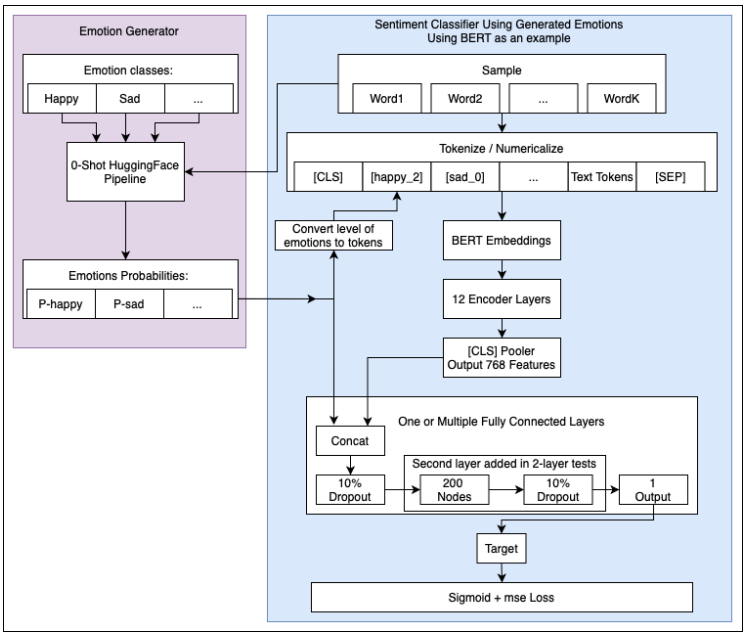

Figure 1: Tokenized text using custom emotion tokens. Then the emotion probabilities are concatenated with the transformer output and processed by one or two fully connected layers.

We use the HuggingFace transformer library Wolf et al. (2019) to load a 0-shot pipeline. They use a Natural Language Inference (NLI) method which considers a premise and hypothesis in order to predict whether the hypothesis holds. The pipeline design can use any model, by default they use BART Lewis et al. (2019) which shows a large improvement over BERT on Yahoo Answers, (see the results from this paper comparing BART and BERT for 0-shot classification Yin et al. (2019)). We use this pipeline to compare each emotion class to the sample text individually forcing the HuggingFace 0-shot pipeline to generate a probability independent of other emotion categories. We kept the default hypothesis format "This example is { }" where { } gets replaced with the respective emotion, and the premise is the original sample. Each emotion is processed individually producing a raw probability from BART whether the hypothesis follows the premise using direct probabilities (not processed through softmax). This gives us the emotion probabilities rather than associating a high probability to the emotion that applies the most to the text from a large list of emotions. BART simply processes the text as "[CLS] premise [SEP] hypothesis [SEP]" and produces a probability for entailment. We used 4 Nvidia P-100 GPUs to predict on 1000 samples per batch (depending on length of text, sometimes the batch must be reduced in size for lengthy samples) to generate the probabilities for each emotion category and for each sample.

Therefore, each emotion category received a probability for how much it associates to the original sample text. The probabilities are pre-processed to fit into a desired range of tokens. The token range is a list of custom transformer tokens where each token represents a segmentation of the probability. For example, if "Happy" has a predicted probability of 0.75 for a specific sample, and we want the range squished into a list of 3 tokens, then we generate the token "[happy_2]" where "[happy_0]" represents all probabilities $<$ 1/3, "[happy_1]" represents the remainder of probabilities $<$ 2/3, and "[happy_2]" represents the remaining probabilities $>=$ 2/3. We do this for each sample and for each emotion, then the sample text is prepended with the list of emotions. The entire token is associated to one embedding, so there's no meaning behind the format aside from organization. For example, the text would look like the following (assuming the emotion ranges): "[CLS] [amusement_1] (...) [happy_2] (...) [worried_0] The original sample text. [SEP]". The emotions prepended allows them to be kept in the same position for all samples for the positional embedding.

In addition, we tested binary labels for the generated emotions using a dynamic threshold (optimal to split the dataset evenly) for each emotion. For a particular emotion, the threshold is determined by sorting the probabilities of the training set and choosing the median of the sorted list as the threshold

Table 1: Comparison to original BERT. IDs describe the types of tests: O := original transformer, T := includes tokens, C := concatenated emotions to output layer, M := mix of T and C, and EN := ensembles. Bold results are all of the models that contribute to EN. EN+Distil include DistilBERT (from Table 2 in bold) with the EN models. Underlined are the result with a P-value lower than alpha=0.10. Columns in order: Test method-ID#, number of output layers, # of tokens used, concat probability format, test accuracy, F1, F1 difference to original, and finally P value.

ID	Classifier Layers	Tokens	Concat Style	Accuracy	F1	Improvement over O-1	t-test P-value
O-1	1	–	–	0.87418	0.87413	N/A	N/A
O-2	2	–	–	0.87373	0.87347	-0.066%	5.75e-01
T-1	1	2 (opt)	–	0.87597	0.87598	0.185%	2.08e-02
T-2	1	2	–	0.87581	0.87546	0.133%	4.53e-09
T-3	1	10	–	0.87615	0.87584	0.171%	1.75e-15
T-4	1	20	–	0.87662	0.87651	0.238%	4.12e-02
T-5	1	30	–	0.87646	0.87629	0.216%	4.67e-01
T-6	2	20	–	0.87652	0.87677	**0.264%**	2.84e-08
T-7	2	30	–	0.87627	0.87632	**0.219%**	6.89e-09
C-1	2	–	(0-100)	0.87611	0.87591	**0.178%**	6.75e-01
C-2	1	–	(1/0) (opt)	0.87604	0.87583	0.170%	7.42e-02
C-3	2	–	(1/0) (opt)	0.87683	0.87650	**0.237**%	5.31e-01
M-1	2	2 (opt)	(1/0) (opt)	0.87652	0.87611	0.198%	5.73e-10
M-2	2	20	(0-100)	0.87624	0.87628	**0.215%**	4.60e-15
M-3	2	20	(1/0) (opt)	0.87604	0.87604	0.191%	1.58e-06
EN	–	–	–	0.87896	0.87890	0.477%	1.38e-04
EN+Distil	–	–	–	0.87922	0.87925	0.512%	8.59e-09

to evenly split the dataset. For example, the generated emotion *empty* has the majority of the dataset classified as one label (97.89% of the dataset). This is uninformative to the model, so instead we apply the dynamic threshold and have 50% of the samples predicted as empty. A high performing emotion such as *unhappy* splits the dataset nearly in the middle at 51.14%, meaning the emotion is more informative to the model. The dynamic threshold in this case will have little change to these emotions when adjusted to a 50.0% split (only 1.14% of the labels change classes). This has two benefits, the reduction of tokens (and added embeddings) when we insert tokens into the text, and the reduction of class imbalance and inconsistencies when training. The first benefit reduces the number of added embeddings to 2 embeddings (a range of 2, positive and negative) per emotion compared to several tokens per emotion depending on the prior emotion level, but maintaining a similar improvement in F1.

Note that we've tried adding the emotions as keywords to avoid training additional tokens and adding negation for positive and negative classes such as "happy" and "not happy" (as text, not tokens), or simply not including the emotion when negative. This did not improve results in testing in any trial. We believe this could be from the positional embedding since each keyword is split into one or many tokens during the tokenization process, and secondly from an association to seeing keywords like "happy" as a biased indication of positive sentiment when possibly appearing in the sample text itself. In this case, using added embeddings allows the model to interpret its own meaning to the emotion while reducing the lengthy input size to each sample using raw keywords.

We use the python libraries Transformers Wolf et al. (2019) and FastAI Howard et al. (2018) to load the pretrained BERT transformer including the vocabulary and zero-shot pipeline. Using FastAI, we tokenize the entire training, validation and test set. Finally, the data collection from FastAI is loaded using the sample text with embedded emotion tokens, sentiment targets, final classifier emotion inputs.

Table 2: Comparison to original DistilBERT. Columns follow the same format as table 1. The results in bold contribute to the ensemble of DistilBERT models. Underlined are the result with a P-value lower than alpha=0.10.

ID	Classifier Layers	Tokens	Concat Style	Accuracy	F1	Improvement over Original	t-test P-value
Original	2	–	–	0.86870	0.86863	N/A	N/A
Tokens	2	20	–	0.87221	0.87263	**0.400%**	1.18e-20
Concat	2	–	(0-100)	0.87217	0.87226	**0.363%**	1.80e-03
Mixed	2	20	(0-100)	0.87218	0.87231	**0.368%**	1.76e-04
Ensemble	–	–	–	0.87479	0.87505	0.642%	1.48e-11

3.3 Incorporating Generated Emotions Into Transformers

In this section we describe the model designs which incorporate the generated emotion information. We specifically tested our models using BERT and DistilBERT, but the approaches can be used on any transformer architecture. We use the following optimization parameters: momentum set to (0.8, 0.7), discriminative weight decay (1e-7, 1e-5, 1e-4, 1e-3, 1e-2) segmented by the following components (embedding, encoder layers 0-3, encoder layers 4-7, encoder layers 8-11, pool and classification layers), and discriminative learning rates set to a range of 8e-6 to 1e-4 for the segmented layers of the model. We use FastAI's fit function to take advantage of these optimization parameters to train the models over 2 epochs. The optimization parameters were tuned to have the model fit by the end of 2 epochs, one more epoch would begin to overfit the model slightly. We split DistilBERT in a similar fashion, but using 2 encoder layers per section rather than 4.

The first step in processing the samples with emotions is prepending the emotions as a text input to the transformer. We altered the tokenizer to expand the vocabulary to include the pre-processed custom tokens like "[happy_2]" as one whole token. Then the transformer is resized to fit this vocabulary with a newly initialized embedding for each custom token. Finally, the text with prepended custom tokens can be tokenized and processed as whole tokens by the transformer. Note that we test the alternative labeling using optimal thresholds (1/0 binary classification of emotions) in place of the range of tokens and/or classifier inputs.

As an additional experiment, we input the probabilities of each emotion category for the sample into the classification layers of the transformer. The classification layers will have the transformer output appended with the emotion probabilities, then one or two fully connected layers with 200 neurons and finally an output layer for the sentiment probability. At prediction time, we can use a threshold of 0.5 to determine positive or negative sentiment. See Figure 1 for a diagram of the full model.

4 Experimentations and Results

We compared different combinations of emotion embeddings to the original BERT transformer without any additional information (see table 1). We trained all of our models on the same data using a test size of 20%, a validation size of 12% and a training size of 68%. The table shows five sections, a section 'O' for the original BERT models (either 1 or 2 classification layers), a section 'T' with added custom tokens to the transformer input only, a section 'C' with emotions concatenated to the classification layer(s) only, a section 'M' where we try a combination of concatenation and custom tokens, and finally an ensemble. The fifth section is an ensemble of all models highlighted bold in their improvement column, and another ensemble (ID EN+Distil) which includes the same BERT models and additional DistilBERT models trained with emotions highlighted in bold in table 2. We also compare the DistilBERT models results on the same problem using the best emotion inputs we found for BERT (see table 2). We include the same five sections as in table 1 and the ensemble is all DistilBERT models highlighted in bold.

We tested each model using Accuracy and F1, and show the difference (improvement) over the original BERT (table 1) or DistilBERT (table 2) model. We include a t-test P-value score for each model, note that the 2 layer original BERT (ID O-2) has a P-value of 0.57459 much greater than alpha=0.10

Table 3: Results for the Sentiment140 dataset using the 0-shot HuggingFace pipeline where we use the predictions of the individual emotion to classify the dataset. Highlighted in bold are direct positive and negative sentiment predictions and their softmax result.

Emotions	Acc	Emotions	Acc	Emotions	Acc	Emotions	Acc
unhappy	0.7813	desperate	0.7195	offended	0.6393	revulsion	0.5758
pos/neg soft.	**0.7715**	contrary	0.6999	joy	0.6367	nauseated	0.5721
disappointed	0.7667	disapproving	0.6966	excitement	0.6290	anger	0.5707
fun	0.7664	miserable	0.6933	withdrawal	0.6288	happiness	0.5692
grieved	0.7649	heartbroken	0.6906	stressed	0.6206	contentment	0.5666
frustrated	0.7601	nervous	0.6823	insulted	0.6185	lost	0.5656
disturbed	0.7575	anxious	0.6822	hopeless	0.6181	love	0.5572
uncomfortable	0.7545	doubtful	0.6800	disgust	0.6169	fear	0.5458
annoyed	0.7514	mad	0.6784	resigned	0.6164	compassion	0.5448
irritated	0.7471	infuriated	0.6713	relief	0.6159	peace	0.5429
worried	0.7465	panicked	0.6695	lonely	0.6157	pride	0.5426
peeved	0.7460	aversion	0.6640	confused	0.6090	surprise	0.5394
troubled	0.7451	enthusiasm	0.6547	terrified	0.6082	hate	0.5338
negative	**0.7377**	satisfaction	0.6508	amusement	0.6013	boredom	0.5224
positive	**0.7333**	bitter	0.6503	cheated	0.6007	empty	0.5132
gloomy	0.7306	sadness	0.6450	loathing	0.5899		
dislike	0.7223	horrified	0.6436	vengeful	0.5826		

Table 4: Tests on small training sets randomly selected from Sentiment140. The samples columns has the number of training samples used. The F1 score is provided for each test, one without the use of tokens and the other with tokens using a range of 20.

Samples	F1 original	F1 tokens	Improvement
50000	0.84347	**0.85081**	0.734%
25000	0.83519	**0.84548**	1.029%
10000	0.82279	**0.83341**	1.062%
5000	0.81387	**0.82338**	0.951%
1000	0.79162	**0.80036**	0.874%
750	**0.78456**	0.78335	-0.121%
500	**0.78449**	0.78039	-0.410%

when compared to the one layer model (ID O-1), while also keeping almost no change in accuracy and F1, meaning we cannot reject the Null Hypothesis, which is expected. Other models have a higher P-value than alpha which are models using too many tokens (30 tokens for example) or using just classifier inputs, but it is possible that we do not have enough information about these models to show a significant change for P-value since accuracy and F1 improve. This is in-line with the fact that the 2 layer BERT model does not show much difference in terms of accuracy and F1 compared to the 1 layer model. All other models show P-values much lower than alpha=0.05 meaning we can reject the Null Hypothesis for the models embedded with additional emotion information. Thus embedding emotions into the models (using tokens and/or concatenation) generates improved and differing results from the original model. Additionally, adding the predictions of each model as an ensemble shows an improvement by combining these differing models.

We found that adding emotions regardless of the input style has a statistically significant improvement over the original model for both cases (BERT and DistilBERT). The best improvement was achieved using the 20 token range input style with a modest improvement of 0.264% and 0.400% for BERT and DistilBERT, respectively. The concatenation of emotion probabilities does not improve nearly as much as with the token models, but we see an improvement over the original models. Using optimal thresholds show a greater improvement within the concatenation tests, but not for tokens inputs. The attempt at mixed tests (tokens inputs and concatenation) show an improvement as well,

Table 5: Tests performed by reducing number of emotions for the Sentiment140 dataset. We used a token range of 20 for these tests.

# Emotions	F1	Difference
63	0.87677	0.264%
31	0.87630	0.217%
15	0.87557	0.144%
0	0.87413	N/A

but averaged out between the improvement provided by tokens and concatenation individually. Additionally, the ensemble of BERT models shows an improvement of 0.477% over the original BERT. The ensemble of the three DistilBERT models using added emotions improves over the original BERT model by 0.092% while improving the original DistilBERT model by 0.642%. Finally, the ensemble of the selected BERT and DistilBERT models show our highest improvement of 0.512% over the original BERT model. The runtime to train each model ranged from 3 to 4 hours on 4 Nvidia P-100 GPUs and a batch size of 100.

We also use the optimal threshold for each emotion to classify the whole dataset using the 0-shot model. We present the results of each individual emotion used to classify the dataset as positive and negative sentiment (see table 3). We also included an attempt to classify as positive and negative sentiment directly by using the 0-shot model and words "positive" and "negative" as classes. We included a softmax classification between both categories as well shown as "pos/neg soft." in the table. The results of this test shows that none of 0-shot predictions comes close to the accuracy of the original BERT or DistilBERT models for direct sentiment classification. If one of the 0-shot predicted emotions could classify the dataset with better accuracy, then the model could rely on that input for its improvement in classification results. Instead, the models use the emotion information for improved sentiment classification since the best performing emotion is "unhappy" with 78.13% accuracy, roughly 8-9% below BERT or DistilBERT's performace. Even a direct prediction with positive and negative sentiment using softmax ("pos/neg soft.") achieves just 77.15%, lower than the "unhappy" emotion result.

We gathered a list of emotions selected from the top of table 3 which most closely resembles the final target (positive/negative) using the top 31 and 15 emotions in an attempt to cut the list of emotion inputs down by half (for example the 15 selected were: 'desperate', 'dislike', 'gloomy', 'troubled', 'peeved', 'worried', 'irritated', 'annoyed', 'uncomfortable', 'disturbed', 'frustrated', 'grieved', 'fun', 'disappointed', 'unhappy'). We present the results in table 5 which shows the reduction of the number of emotions (even chosen specific for sentiment) correlates to the reduction of accuracy using the best 20-token model.

We also show the variation of the results of our best 20-token BERT model by reducing the training set size. Note that the size of the test set remains the same as in all other tests. We show in table 4 that our design benefits smaller datasets more significantly than larger sets at a range of 5000-25000 training samples. Even as low as 1000 samples we can show a modest improvement until we lose accuracy at 750 and 500 samples.

To compare with other approaches applied on the same sentiment140 dataset, Tago et al. (2019) use a selection strategy to use dates and user information along with emoticons to classify the dataset with 77.6% accuracy. In Wang et al. (2020) authors report accuracies of several models using a 70/30 train-test split for the following models: 82.00% multilayer LSTM using embeddings, 77.98% SVM, 76.44% Logistic Regression, 76.52% Multinomial Naive Bayes, and 77.67% using an ensemble. In all approaches, models have a difficulty achieving high accuracy results with minor differences in results without using complex models such as the multilayer LSTM with embeddings. Imran et al. (2020) claim to have state-of-the-art (SOTA) with 0.824 F1 on the dataset using an LSTM on a 10% test set outperforming previous SOTA using a convolutional neural network (CNN) with a variety of embeddings including BERT embeddings. We currently have 0.87925 F1 using an ensemble while also achieving 0.87677 F1 from a single BERT model using 20% test set, 12% validation and 68% training.

Table 6: US Airline results using original BERT (ID O), BERT using the 2 token approach (ID T), BERT using concatenated classifier inputs (ID C), and BERT using a mix between the 2 token approach and concatenated classifier inputs (ID M). The "Improv." column is the improvement in F1 over the original BERT score. Underlined are the result with a P-value lower than alpha=0.10.

ID	Accuracy	F1	Improv.	t-test P-value
O	0.93869	0.93830	N/A	N/A
T	0.95032	0.95079	1.249%	2.86e-02
C	0.95032	0.94930	1.100%	6.87e-02
M	0.94609	0.94557	0.727%	7.06e-01

For the US Airline sentiment dataset, Table 6 presents our results for the original BERT, BERT with the 2 token approach, BERT with concatenated classifier inputs, and BERT with a mix of tokens and concatenated inputs, all using 2 classification layers. We see an improvement of 1.249% with 0.95079 F1 in the 2 token approach over the original BERT. With nearly 5000 training samples in this dataset, the runtime to train each model took roughly 5-10 minutes on 4 Nvidia P-100 GPUs. For comparison with other approaches, Rani & Gill (2020) achieve 87.422% accuracy using an ensemble of Naive Bayes, svmRadial, C5.0 and a DictionaryBased Classifier. Umer et al. (2020) use a CNN-LSTM combined model with 82.0% accuracy and 83.0 F1. Note that these comparisons use the neutral samples which were omitted from our tests.

Authors of TweetBERT Qudar & Mago (2020) have shown a great improvement in tweet classification, especially in the Sentiment140 and US Airline datasets we've tested with in this paper. The authors use BERT in its current state and continuously pretrain on a very large amount of data including 680 million tweets. The continuously pretrained BERT presents a much better understanding of tweets compared to BERT with reported accuracies of 95.18% and 92.99% on Sentiment140 and US Airline, respectively. Since our approach is general for any transformer and tested for BERT, emotions embedded as tokens for this pretrained BERT model could likely benefit from our approach and improve the Sentiment140 benchmark further.

5 Conclusion and Future Work

In conclusion we show modest improvements on BERT and DistilBERT by embedding emotions in both custom tokens and classification layer inputs. Our best single model with added emotions achieves 0.87677 F1 using BERT. We see an improvement of 0.264% in BERT using a custom token range of 20 while DistilBERT improves by 0.400% using the same approach. Using an ensemble of BERT models using emotion inputs we can achieve an improvement of 0.477% and similarly an improve of 0.642% for DistilBERT models over the original DistilBERT model and a 0.092% improvement over the original BERT. Finally we get a total improvement of 0.512% over the original BERT using an ensemble of BERT and DistilBERT models. We additionally test our approach on the US Airline dataset and show how emotions using the 2-token approach improves F1 by 1.249% to 0.95079 F1. While the observed improvements are modest, they are systematic, each method we used to add additional information outperformed the original BERT classifier. However, most of our model configurations tested shown a statistically significant improvement over a direct classification without additional information. These results are therefore promising to showing that it could be possible to improve accuracy of most text classification tasks by adding additional semantic information generated using 0-shot learned on pre-trained models. The main interests of our approach are that it is independent from the model used, simple to apply and systematically improve the accuracy/F1 scores, even using a small training set. In the future, we plan to continue research using 0-shot models and the possibility to incorporate more 0-shot information in our classification models. We plan to try our approach on other sentiment datasets and to expand datasets with additional data and labels in an automatic system to improve general article classification tasks beyond sentiment classification.

REFERENCES

Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*, 2020.

Md Arid Hasan, Jannatul Tajrin, Shammur Absar Chowdhury, and Firoj Alam. Sentiment classification in bangla textual content: A comparative study. *arXiv e-prints*, pp. arXiv–2011, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Jeremy Howard et al. fastai. `https://github.com/fastai/fastai`, 2018.

Ali Shariq Imran, Sher Muhammad Daudpota, Zenun Kastrati, and Rakhi Batra. Cross-cultural polarity and emotion detection using sentiment analysis and deep learning on covid-19 related tweets. *IEEE Access*, 8:181074–181090, 2020.

Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. Supervised multimodal bitransformers for classifying images and text. *arXiv preprint arXiv:1909.02950*, 2019.

Nikita Kitaev, ukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019.

Usman Naseem, Imran Razzak, Katarzyna Musial, and Muhammad Imran. Transformer based deep intelligent contextual embedding for twitter sentiment analysis. *Future Generation Computer Systems*, 113:58–69, 2020.

Daniel Palomino and Jose Ochoa-Luna. Palomino-ochoa at semeval-2020 task 9: Robust system based on transformer for code-mixed sentiment classification. *arXiv preprint arXiv:2011.09448*, 2020.

Mohiuddin Md Abdul Qudar and Vijay Mago. Tweetbert: A pretrained language representation model for twitter text analysis. *arXiv preprint arXiv:2010.11091*, 2020.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Sangeeta Rani and Nasib Singh Gill. Hybrid model using stack-based ensemble classifier and dictionary classifier to improve classification accuracy of twitter sentiment analysis. *International Journal*, 8(7), 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2019.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404. URL https://www.aclweb.org/anthology/D18-1404.

Kiichi Tago, Kosuke Takagi, and Qun Jin. Polarity classification of tweets considering the posters emotional change by a combination of naive bayes and lstm. In *International Conference on Computational Science and Its Applications*, pp. 579–588. Springer, 2019.

Muhammad Umer, Imran Ashraf, Arif Mehmood, Saru Kumari, Saleem Ullah, and Gyu Sang Choi. Sentiment analysis of tweets using a unified convolutional neural network-long short-term memory network model. *Computational Intelligence*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Diangang Wang, Lin Huang, Xiaopeng Lu, Yan Gong, and Linfeng Chen. Research on text sentiment analysis based on attention c_mgu. In Jing Liu, Honghao Gao, Yuyu Yin, and Zhongqin Bi (eds.), *Mobile Computing, Applications, and Services*, pp. 163–173, Cham, 2020. Springer International Publishing. ISBN 978-3-030-64214-3.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019. URL http://arxiv.org/abs/1906.08237.

Da Yin, Tao Meng, and Kai-Wei Chang. Sentibert: A transferable transformer-based architecture for compositional sentiment semantics. *arXiv preprint arXiv:2005.04114*, 2020.

Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach, 2019.

David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. *ACM Transactions on Management Information Systems (TMIS)*, 9(2):1–29, 2018.