University of Pisa
Text Analysis
A.Y 2024-2025

Emotion Detection on Remote Working

Submitted to:
Laura Pollacci PhD

Submitted by:
Sara Hoxha
Rafael Ignacio Urbina Hincapie
Helen Naomi Cedeno Manrique
Luis Hernan Pinto Aleman

January 12, 2025

Contents

1 Abstract

This study uses machine and deep learning models to explore emotion detection in remote work discussions. Social media data was processed and classified into six emotion categories based on Parrott's "Basic Emotions" framework. Performance of models such as SVM, RNN, BiRNN, and LSTM was evaluated. Explainability was integrated via LIME to give insight into the working of these "black-box" models. Moreover, the NRC Emotion Lexicon provided a traditional benchmark for our models. The results emphasize the potential of deep learning and explainable AI to understand emotional trends, offering valuable insight into the impact of remote work.

Keywords: Emotion detection, remote work, machine learning, deep learning, explainable AI, social media analysis.

2 Introduction

Since the COVID-19 pandemic, working from home (WFH) has become the new way of working for millions of employees in the EU and around the world. The increasing of remote work has changed the way people work and live. On the positive side, remote work gives employees more flexibility, reduces mobility time and allows for greater control over their schedules [Felstead and Henseke, 2017]. However, it also presents some challenges, such as increased stress, depression, and can fade the lines between work-life boundaries [Oakman et al., 2020]. This shift to remote work has undeniably sparked a broad spectrum of emotions among employees, both positive and negative. Therefore, understanding the emotional landscape of remote work is key for companies to retain talent and meet employee needs.

Our goal is to gain insights and shed light on hidden patterns into how remote work is impacting the workforce. We aim to develop an emotion model that detects emotions from social media posts and discussions where remote workers share their experiences. The model will use the "Basic Emotions" framework as developed by psychologist [Parrott, 2001], which includes the following recognized emotions: love, joy, surprise, anger, sadness, and fear.

The entire project was developed in Python. In the first stage, we collected data from Reddit social network after the COVID period. Next, we used pre-trained models, T5 and Yangswei, to generate the ground truth by aligning a range of emotions with Parrot's system. After that, we performed a pre-processing step, which included text cleaning, tokenization, stopword removal, and emoji mapping. Then, we moved on to model selection and training. The models we chose for emotion classification were RNN, BiRNN, LSTM, and SVM. Additionally, we used explainability algorithms (XAI) to make the classification process easier to understand and address the "black box" problem. Finally, we used the NRC Lexicon to predict emotions in the dataset and conducted a comparative analysis with the two earlier approaches.

3 Methods

3.1 Data Collection

Initially, two social networks were used to collect posts related to remote work. Namely: X (formerly known as Twitter) and Reddit. To identify such posts, the search was conducted using

the following hashtags: *workfromhome*, *remotework*, *smartwork* and *remoteworking*. On Reddit, on the other hand, information was obtained from the subreddits: *remotework*, *workfromhome* and *remotejobs*. However, after an exhaustive analysis of the results of both collection processes, it was decided to use only the information from Reddit, since on Twitter the limitation of the number of characters per post, the access to the API of this social network, especially on the access to historical data, among other circumstances, may compromise the quality of the data extracted. Based on the above, 175,148 posts were collected based on the parameters indicated above. From each post, it was possible to extract:
1. `topic_title`. The main topic of the post.
2. `text:` The full text of the publication in question.
3. `author:` The username of the author of the post.
4. `subreddit:` The name of the subreddit to which the post belongs. 5. `num_comments:` The number of comments on the given post.
6. `upvotes:` The number of upvotes for the post.
7. `created_utc:` The creation date of the post.
It is important to mention that, to extract the posts from Reddit, we used the packages `praw` [Boe and Payne, 2023] and `pandas` [Wes McKinney, 2010]. In addition, only posts created after the COVID-19 pandemic were considered.

3.2 Ground-Truth creation - Pretrained models

In order to label each observation with a particular emotion and identify them as *ground truth*, we searched for pre-trained models available in *HuggingFace*. Thus, we highlight two models: Google's T5 [Raffel et al., 2023] model trained for emotion detection and Yangswei's distilBERT-based model. These models label texts within one of the six primary Parrot [Parrott, 2001] emotions: sadness, love, joy, anger, fear and surprise. Both models present an accuracy of 0.93, as well as a recall and F1-score greater than or equal to 0.90 in their pre-training process. In addition, both the T5 model and the Yangswei model have been pretrained from the same dataset. Therefore, these models are used for their power to predict and classify emotions from text, as well as for their alignment with Parrot's emotion tree. To ensure the quality of the outputs generated by the models, a sample was generated to perform a logical check of the alignment of the emotion with the text itself.

3.3 Data Preprocessing

The collected information was processed to be adapted as input for machine learning methods, dividing the preprocessing stream into general and specific steps according to the prediction technique applied. Initially, Twitter and Reddit data were handled with separate cleaning functions, but when removing tweets, the functions were consolidated into single ones.

First, the `text` feature was cleaned, converting all of the characters to lowercase, removing also the links and editing paragraphs that are usually corrections without emotional relevance and repeated spaces. In addition, only alphanumeric characters, question marks, exclamation marks, and money symbols were retained, the latter because their role is considered important in posts expressing opinions on home office salaries. It is important to mention that all non-alphanumeric characters were mapped as words afterward.

Next, comments with less than 30 characters, which are generally job offers or advertisements that do not contribute sentiment, and posts with negative votes were removed, since on Reddit these usually indicate no relevance or contribution to the main topic of the sub-

reddit. Also, the title was added at the beginning of the text, aiming to improve the context and reduce ambiguity in posts that are comments. Then, emojis were removed to simplify the analysis since they are not commonly used in Reddit and, when they are, they usually reflect sarcasm or exaggerations that can complicate the interpretation, Moreover, pre-trained models are generally not optimized to handle emojis effectively.

Finally, the length distribution of the texts was analyzed to ensure that it did not negatively influence the label generation, keeping only the observations within the 85th percentile, and consequently eliminating texts that were excessively long.

3.3.1 SVM

For the SVM model specifically, we needed to clean the data by doing the tokenization, stop words removal, and lemmatization, this was made using both nltk [Wagner, 2010] and spacey libraries [Honnibal and Montani, 2017], having selected English as the language.

After this first phase of preprocessing, we used a vectorizer to organize the words on a matrix of occurrences. With this, we can feed our model, giving each word a different importance.

A second approach for the preprocessing was to try a word embedding representation for the SVM, this one is mentioned and explained in the SVM modeling section.

3.3.2 Deep Learning Models

In the LSTM, RNN, and BiRNN models, a set of general utilities was applied, starting with expanding the contractions within the text, e.g. converting "don't" to "do not", to enhance the consistency and coherence of the text.

Next, similar processes were applied to prepare the input data for the models, but with some slight differences. For the LSTM, a single `TensorFlow/Keras` [Abadi et al., 2015] codebase was implemented over the dataset, applying the `tokenize_and_pad` function to convert the text into numerical sequences through tokenization and adjust them to a uniform length using padding. The output consisted of input sequences, along with the generated vocabulary and its corresponding size. Then, the labels were encoded into numerical values using a `LabelEncoder` and subsequently transformed into a one-hot format with `to_categorical`, allowing an adequate multi-class classification in the model.

On the other hand, both the RNN and BiRNN followed a parallel preprocessing workflow as the LSTM but divided into training and testing phases. In this case, the tokenizer used for the training set was saved as a `.pickle` file to reuse it in the test set. Additionally, this code was implemented in `PyTorch` [Paszke et al., 2019], transforming the labels into tensors and storing the class information in a `.npy` file to ensure consistency in the encoding.

3.4 SVM

Two distinct approaches were employed for SVM implementation. Initially, a pipeline involving sentence splitting, stop-word removal, and CountVectorizer was implemented to extract feature vectors from the text data. Grid search was then utilized to optimize hyperparameters, including the number of features selected by SelectKBest and the regularization parameter 'C' on SVM. Additionally, a TfidfTransformer was applied to assign appropriate weights to the input data. This initial approach achieved moderate success, with accuracies reaching up to 77%. While the F1-score varied across classes, most classes exhibited scores above 0.50, with 'joy' reaching 0.85. However, the limited dataset size likely constrained the model's performance, particularly for classes with lower representation, such as 'love' and 'fear'.

To enhance performance, two strategies were explored. First, **class balancing** through weight adjustments during SVM prediction. However, this approach consistently yielded lower performance compared to the unweighted model, possibly due to the limited dataset size and potential overfitting of the weighted model.

Secondly, the SVM model was fed with pre-trained **word embeddings** instead of raw text and TF-IDF features. Pre-trained models such as **GloVe.6B.300d** (trained on 6 billion tokens from Wikipedia and Gigaword) and **FastText** (leveraging subword information for improved out-of-vocabulary word handling) were evaluated. **FastText** consistently outperformed **GloVe** in this context. A **custom Word2Vec** model trained on the dataset was also explored but exhibited inferior performance. While using **FastText** embeddings with different SVM kernels improved results, they did not surpass the performance of the initial approach using CountVectorizer and TF-IDF.

3.5 Deep Learning Models

We evaluated three neural network architectures — Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Bidirectional Recurrent Neural Network (BiRNN) — to address the classification task. The input text data underwent a comprehensive preprocessing pipeline, including the expansion of contractions, conversion of text to lowercase, and removal of non-alphanumeric characters. To standardize input lengths across the models, tokenization and padding were performed using TextVectorization from `Tensorflow` library [Abadi et al., 2015], and class labels were transformed into numerical representations using the LabelEncoder from `scikit-learn` library [Pedregosa et al., 2011].

3.5.1 RNN & BiRNN

We programmed all the recurrent neural networks using the `Tensorflow-Keras` library [Chollet et al., 2015]. In regards to our implementation approach, both models shared a foundational structure with key differences in their architecture:

- **Recurrent Neural Network (RNN)**: Comprised of an embedding layer, one or two `SimpleRNN` layers, each with 32 to 128 hidden units, followed by a `softmax` output layer for classification. Dropout layers were included after each layer to mitigate overfitting, with L2 regularization applied to the weights.
- **Bidirectional Recurrent Neural Network (BiRNN)**: Extended the RNN architecture and aimed to improve by wrapping a `LSTM` layer, instead of a `SimpleRNN` one, with a `BiDirectional` wrapper, allowing the network to process sequences in both forward and reverse directions, capturing context from both ends of the input sequence.

Hyperparameter optimization was performed for both architectures using Keras Tuner's Random Search, varying parameters such as embedding dimensionality, the number of layers, units per layer, and dropout rates. The tuning objective was to maximize validation accuracy, with up to 25 trials and two executions per trial.

The models were trained using the Adam optimizer and categorical cross-entropy loss function. A batch size of 64 was used, and training was conducted for a maximum of 50 epochs. Early stopping and model checkpoint callbacks were employed to save the best-performing model based on validation loss and to prevent overfitting by halting training when validation performance plateaued for 15 consecutive epochs.

Ultimately, the best configurations from each NN model, identified during the training phase, were evaluated on the test set. The preprocessing of the test data mirrored that of the training set, utilizing the same tokenizer and label encoder. To assess the classification performance, a myriad of metrics and plots were used which we delve in further in the next section.

3.5.2 LSTM

The Long short-term memory model (LSTM) was built using the `Tensorflow-Keras` library [Abadi et al., 2015]. The architecture of the LSTM model includes one embedding layer, and two LSTM layers for processing the embedding sequences and capturing temporal patterns. The coexistence of two LSTM layers allows the identification of more complex patterns. Additionally, a Dropout layer was generated after each LSTM layer to avoid overfitting by randomly disconnecting neural network connections. Next, the *softmax* activation function was used, which is ideal in multiclass problems. Finally, the *Adam* optimizer was used, which combines gradient descent methods with adaptive learning rate adjustments, as well as a categorical crossentropy as a loss function.

With the LSTM architecture described above, we applied `Keras Tuner`, which uses random search to find the optimal hyperparameters of the model to maximize the accuracy of the validation dataset. The hyperparameters to be set were the `output` of the embedding layer, `lstm_units` of the first LSTM layer, `lstm_units` of the second LSTM layer and `dropout_rate` of the Dropout layers. The random search used a maximum of 25 trials, with 2 runs per trial, using a `batch_size` of 64.

Once the optimal hyperparameters for the model were found, the model was run with an early stopping condition, to avoid overfitting problems, where the loss of the validation dataset is monitored, with a `patience` of 3.

3.6 Explainability

3.6.1 SVM LIME

Regarding the explainability methods for the SVM models we implemented, we decided to use LIME [Ribeiro et al., 2016] as our explainer, this method focuses on explaining the prediction for a single data point, it works by approximating the black box model's behavior in the vicinity of the specific data point with a simple, interpretable model.

With the way the explainer works, we decided to sample some records with different labeled emotions to see what the difference is in the terminologies used for the model to classify as emotion A or B. Specifically, we analyzed: Anger, Joy, and Surprise.

3.6.2 NN LIME

To shed light on how black-box deep learning models make their classification decisions, we've decided to delve into its inner workings using LIME (Local Interpretable Model-agnostic Explanations) for both datasets. We have inspected one single randomly chosen instance from our test set, at index 1693. For the Yangswei_58 dataset, this instance had a true class value of "sadness", whereas for T5 the label was "anger".

We decided to explore all three deep learning models (RNN, BiRNN, and LSTM) to better understand their decision-making process and be able to compare not just through metrics but also through explainability.

LIME was conducted on all models for both datasets and the explainability notebooks were saved into an HTML format for easy visualization and further analysis which we conduct in the next section.

3.7 NRC Lexicon

We utilized the NRC Emotion Lexicon, as introduced by [Mohammad and Turney, 2010], a comprehensive list of 13,872 English words associated with eight basic emotions defined by [Ekman and Friesen, 1971]: anger, fear, anticipation, trust, surprise, sadness, joy, and disgust. This lexicon was applied to predict emotion labels for our datasets to compare the performance of traditional lexicon-based methods with our developed models.

For each word in the text, we calculated its emotion scores using the lexicon and assigned the emotion with the highest score as the label. Sentiments labeled as "positive" and "negative" were excluded. To align the NRC results with the output of our models, we mapped the NRC categories to Parrott's emotion framework, which reduced the number of categories from eight to six. This mapping involved the following adjustments:

1. Anticipation to Joy 2. Disgust to Anger 3. Trust to Love

4 Results

4.1 SVM

We got the best parameters of the two best models after implementing hyperparameter tuning, as shown in Table 1. To evaluate their performance, we used a set of metrics, such as accuracy, precision, recall, and F1-Score. Furthermore, we also calculated the confusion matrix to better analyze the results.

Model	**Dataset**	**Parameters**	**Accuracy**	**Weighted Precision**	**Weighted Recall**	**Weighted F1-Score**
LinearSVM, TFIDF, No Class Weights	Yangswei_85	KBest = 4000, C = 3.129812	78%	0.77	0.78	0.76
LinearSVM, TFIDF, Class Weights	Yangswei_85	KBest = 4000, C = 11.356982	76%	0.77	0.76	0.76

Table 1: Performance evaluation of the two best SVM models

The two best SVM classifiers we got from the Yangswei_85 dataset. The results show that the model with no class weights has a higher accuracy (78%) than the model with class weights (76%). Both models have similar weighted precision (0.77) and F1 scores (0.76). This indicates that the performance in terms of precision and recall is almost identical. We analyzed more the confusion matrices to explore why applying the class weights had a slightly negative impact on this dataset. The analysis reveals that while the model with no class weights performed well in the larger classes like "joy" and "anger," the model with class weights improved the accuracy for smaller classes such as "love" and "surprise." However, this improvement in smaller classes led to more misclassifications in the majority classes, which made the overall performance decrease

slightly when class weights were applied. The confusion matrices for both approaches are shown in figure 1.

(a) Linear SVM, No Class Weights

(b) Linear SVM, Class Weights

Figure 1: SVM Confusion Matrix

4.2 RNN & BiRNN

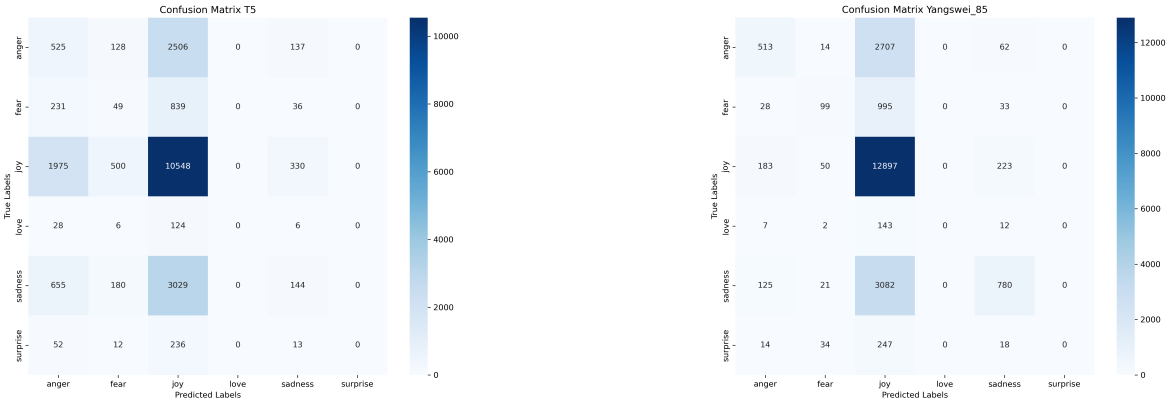
After conducting hyperparameter tuning, the best model architectures we found are displayed in Table 2. As for the evaluation of RNN and BiRNN models, we utilized a comprehensive set of metrics, including accuracy, recall, precision, and F1 score, both weighted and unweighted, alongside additional tools such as classification reports and confusion matrices to thoroughly analyze the models' performance.

Model	Dataset	Architecture
RNN	Yangswei and T5	Output Dimensions: 64 Number of Layers: 2 RNN Layer 1 Units: 32 RNN Layer 2 Units: 64 Dropout Rate: 0.2 L2 Regularization: 0.001
BiRNN	Yangswei	Output Dimensions: 128 Number of Layers: 1 LSTM Units: 64 Dropout Rate: 0.4 L2 Regularization: 6e-3
	T5	Output Dimensions: 128 Number of Layers: 1 LSTM Units: 48 Dropout Rate: 0.4 L2 Regularization: 6e-3

Table 2: Architecture Overview of Best Models for RNN and BiRNN

The RNN model produced unsatisfactory results for both datasets, achieving at most 64% accuracy on the Yangswei dataset compared to 50.55% on the T5. The weighted F1 score was particularly important in addressing class imbalance, ensuring a more comprehensive evaluation

of model performance across all classes. Specifically, the weighted F1 scores were 0.5608 for Yangswei and 0.4491 for T5. The confusion matrix for both datasets can be viewed in Figure 2.



(a) T5 (b) Yangswei_85

Figure 2: RNN Confusion Matrices

As for BiRNN models, we saw that they converged quickly, reaching their optimal performance within just 5 epochs. This rapid convergence suggested that the model was able to efficiently capture patterns in the data, yet it could also indicate overfitting. To mitigate this risk, we included a L2 regularization parameter for the model's weights, as well as utilized dropout. Furthermore, the plot of training and validation losses in Figure 3 revealed that while the training loss decreased rapidly, the validation loss remained slightly higher, and both losses converged around epoch 2.

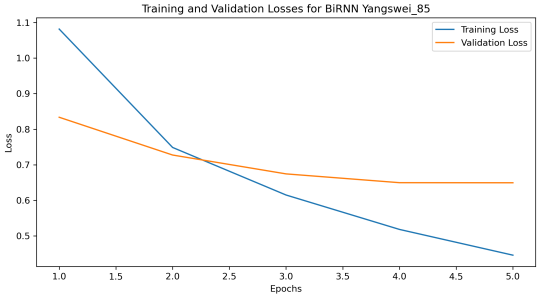


Figure 3: Training & Validation Losses for BiRNN Yangswei_85

BiRNN models outperformed RNN, showing significantly better results across both datasets. For the Yangswei dataset, the BiRNN achieved an accuracy of 78.32% and weighted F1 score of 0.7805. Similarly, on the T5 dataset, the BiRNN achieved an accuracy of 73.24%, with a weighted F1 score of 0.7305. Overall, the model was better at identifying and predicting correct emotions as can be seen in the confusion matrices of Figure 4.

(a) T5

(b) Yangswei_85

Figure 4: BiRNN Confusion Matrices

4.3 LSTM

The results of the hyperparameter tuning determined the following same architecture for both T5 and Yangswei datasets: output_dim:256, number of layers: 2, first LSTM layer units: 256, Dropout: 0.3. The only difference was in the units present in the second LSTM layer were there were 64 units for T5, whereas 96 units for Yangswei. From this setup and subsequent model run, a performance evaluation was performed based on *accuracy*, *precision*, *recall*, *F1-Score*, and the confusion matrix of the results.

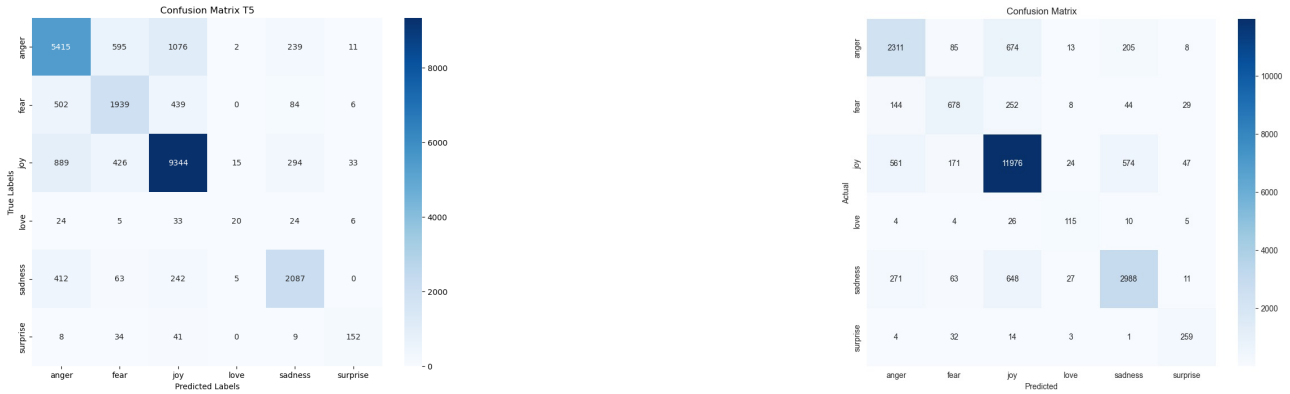
The results show moderate performance, with an average *accuracy* of 0.77, average *precision* of 0.63, average *recall* of 0.70 and average *F1-score* of 0.65 for the T5 dataset. The differences in category performance is mostly due to the imbalance of the dataset, with a higher score for *joy*, which is the highest frequency emotion, while *love*, being the lowest frequency sentiment, presents the lowest performance, in all evaluation metrics.

On the other hand, the Yangswei dataset shows relatively better results. Specifically, the model yields an average *accuracy* of 0.82, an average *precision* of 0.72, average *recall* of 0.74, and average *F1-Score* of 0.73. Class imbalance is less evident in this model, as there is a smaller difference between the evaluation metrics of the highest frequency class (joy), with respect to the lowest frequency class in the data set (love). In fact, in *recall* and *F1-score*, the *love* category is not the one with the lowest scores.

The difference in prediction strength between the different categories can be seen in Figure 5. Here, it can be noticed that the model with the T5 dataset has problems in differentiating between negative emotions, i.e. anger, fear and sadness. On the other hand, the problem of class imbalance is evident, since in low-frequency emotions such as love or surprise, the model does not seem to be able to differentiate between classes.

On the other hand, the model with the Yangswei dataset shows significantly better performance, especially for lower frequency classes such as love and surprise, which may imply that the model is able to achieve better differentiation between classes. However, the classes *anger* and *fear*, suffer a slight decrease in their evaluation metrics with respect to the T5 model.

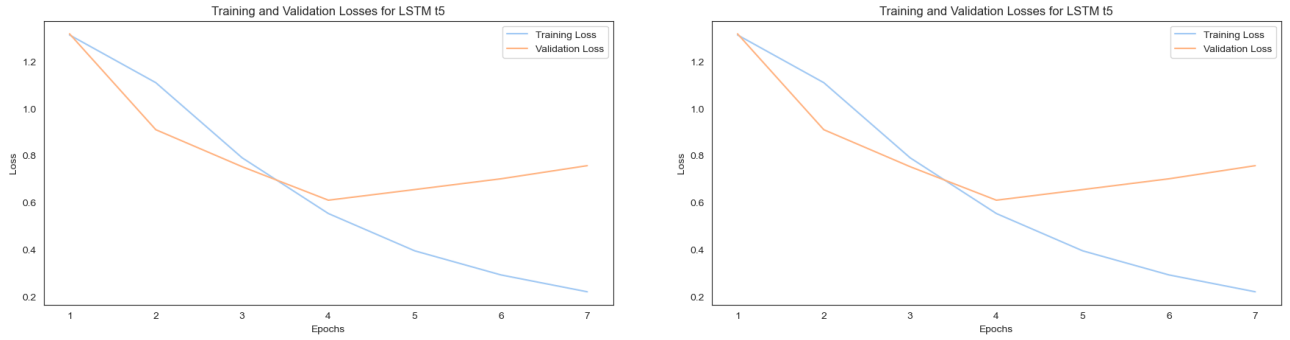
Finally, it can be noted that the training process was relatively short, as less than 10 epochs were required to converge to an optimal result on both data sets. In fact, as observed in Figure 6, the loss, as well as the accuracy of the validation set, is optimized in the fourth epoch, and the difference between the loss of the training set and the validation set is minimal. The learning trend in the T5 and Yangswei data sets is the same.



(a) Confusion matrix T5

(b) Confusion matrix Yangswei

Figure 5: Confusion matrices for the LSTM models: T5 and Yangswei.



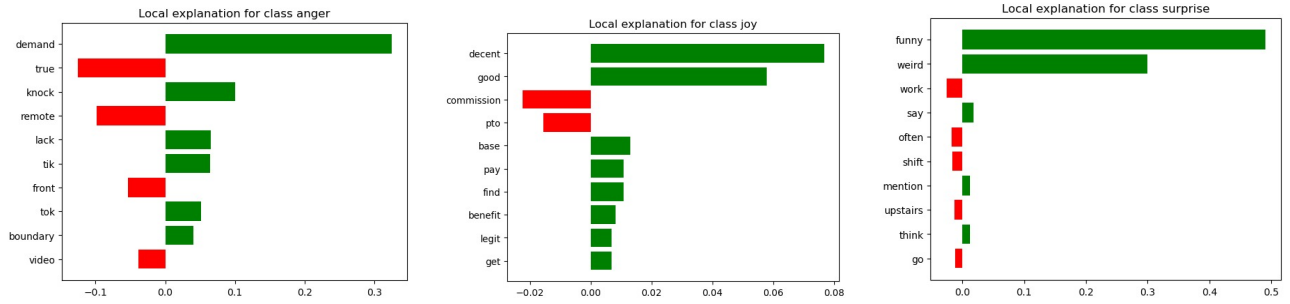
(a) Losses for T5 dataset

(b) Losses for Yangswei dataset

Figure 6: Comparison of losses for LSTM models on T5 and Yangswei datasets.

4.4 SVM LIME

As seen in Figure 7a Lime explainer outlines the top words or characteristic attributes that contribute to the fact that the observation is classified as anger, in this case, we can see that the most important words to classify this post as anger are the words: demand, knock and lack, which can logically be put on a negative context,in turn some of the words that negatively affect the classification of anger are: true and remote.



(a) Anger

(b) Joy

(c) Surprise

Figure 7: SVM Lime explanation for emotion observations

In Figure 7b we can see the top 10 attributes that determine the correct classification of class joy, and to no surprise, we see that some of the words that contribute to the classification

are words such as decent or good.

To conclude we also ran the explainer model for the surprise class in Figure 7c which as the figure shows two of the attributes contain a high amount of the explainability of the classification, if we put this into perspective with the other two explainers shown, we can see how for some observations the explainer can obtain a bigger context of how diverse words affect the classified label, where in other observation the explainer model does not make the best job at explaining how words affect the classification and ends up putting most of the weight to a few words only. The SVM LIME results demonstrate a mix of well-balanced explanations alongside others that disproportionately rely on single attributes, suggesting room for further refinement.

4.5 NN LIME

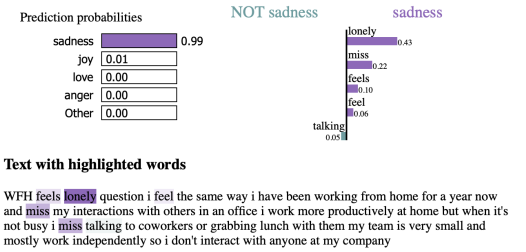
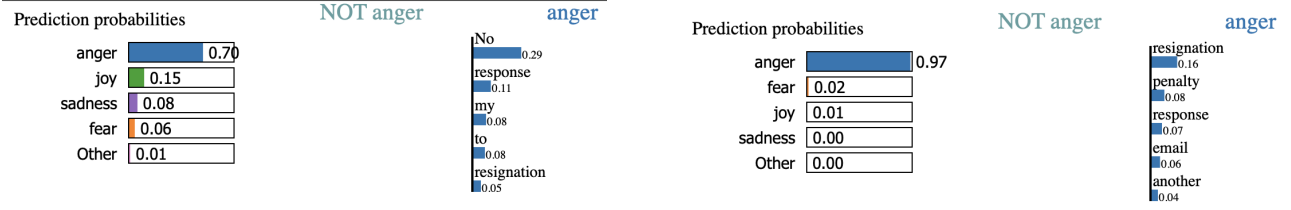


Figure 8: BiRNN Yangswei LIME explanation for instance 1693

LIME was utilized to offer localized insights into the decision-making of the models. By concentrating on the single instance we choose, LIME perturbs the data in its vicinity, observes the model's reactions, and constructs a more simple model specific to that instance. For BiRNN, the explanations given by LIME for instance 1693 of Yangswei and T5 can be seen in Figure 8 and **??** respectively. From the explanation, we can see that our model is accurately able to associate words like "lonely" and "miss" with feelings of sadness and predicts this as the label with a 99% probability. Similarly, for T5 it recognizes "resignation" and "penalty" as related to "anger" with a 97% probability.

Explainability allowed us to observe how our least-performing model (RNN) was making decisions. For example, the text of instance 1693 in the T5 dataset was `"No response to my resignation email question they can be transferred to another k otherwise the early withdrawal penalty would apply"`. As can be seen in Figure 9a RNN was able to classify correctly the instance as "anger" with a not-so-confident 70% probability, yet the "reasoning" is clearly less insightful than that of BiRNN. It is taking into consideration words like "no" and "response" which do not necessarily imply an emotion of anger. Moreover, the fact that it selects "joy" as the second best guess instead of "sadness" shows that RNN is not able to capture the differences between instances. Even though we could deduce some of these insights from the metrics, we found that LIME provided a clear understanding of how even "correctly predicted" instances can be so for wrong reasons.



(a) RNN

(b) BiRNN

Figure 9: T5 LIME explanation for instance 1693

4.6 NRC Lexicon

The NRC-based emotion analysis diverged significantly from those of our deep learning models, as indicated by the comparatively low metrics in accuracy, precision, recall, and F1 score. Specifically, Yangswei achieved an accuracy of 36% with a precision of 0.51 and a recall of 0.36, while T5 performed slightly lower, with an accuracy of 33%, a precision of 0.43, and a recall of 0.33.

5 Discussion

Our findings showed that the majority of emotions related to remote work were positive, predominantly in the "joy" emotion class. For instance, one individual expressed: `"New offer from previous company stay where you are if you're happy with the flexibilit! the effort of going into the office especially as a mother is not worth the time and energy, plus factoring in time to get ready gas money etc if you're enjoying the wfh flexibility they would need to offer way more than k to make it worthwhile"` reflecting a common sentiment of contentment with work-life flexibility and reduced commuting.

As for the performance of our models, overall we obtained satisfactory results and our models were able to capture the patterns in the text provided shown by LIME. For the RNN models, the discrepancy between the results on the two datasets can be attributed to differences in class distributions, vocabulary sizes, and the vanishing gradient problem inherent to RNNs [Hochreiter, 1998]. The Yangswei dataset, with a more severe class imbalance, skewed heavily towards "joy" (59.91% of instances), likely inflated the model's accuracy by favoring predictions for the dominant class. Additionally, the vocabulary size for Yangswei (41,714) was slightly smaller than T5's (44,233), thus making it a simpler model to generalize. In contrast, the T5 dataset, with a relatively more balanced distribution and slightly more complex dependencies posed greater challenges for the RNN. Such observations were the same for the BiRNN model. As expected, the BiRNN models outperformed RNN and addressed the vanishing gradient problem by yielding satisfactory accuracy and weighted F1 score results. The best overall classification results were obtained with the LSTM model, with a *accuracy* of 0.82 and an average *recall* of 0.74. The performance achieved shows the problems that can be caused by the lack of samples of a certain class, as well as the class imbalance of the dataset in general. Moreover, the architecture used proves to be efficient by optimizing its learning in a few epochs, drastically reducing computational time. Overall, the model could be improved with a more exhaustive hyperparameter random search, at the cost of a much longer processing time.

Regarding the SVM implementations, although the results show relatively good performance (76% F1-score) with both datasets, the model could be improved by a further and more specialized grid search, specifically regarding the non-linear kernels, these posed a greater challenge since the fitting of the models was taking several hours per trial, also is important to mention that as per [Machová et al., 2023] the SVM models tend to be outperformed by the NN models in emotion detection tasks with internet posts.

As for the lexicon approach, the unsatisfactory results which did not surprise us, can be attributed to a myriad of factors. Firstly, the NRC lexicon, being a rule-based approach, relies on predefined word-emotion mappings, which may not capture the nuances, and context in our data. Secondly, our necessary process of mapping Ekman's emotions to Parrott's emotion

hierarchy could introduce additional mismatches or inconsistencies, potentially affecting the comparative analysis. Lastly, the lack of domain-specific customization in the NRC approach might result in less relevance to the dataset used, whereas the deep learning models benefited from sophisticated training methodologies tailored to handle such nuances.

Ultimately, the best classification results were obtained with the LSTM model, with a *accuracy* of 0.82 and an average *recall* of 0.74. The performance shows the problems that can be caused by the lack of samples of a certain class, as well as the class imbalance of the dataset in general. Moreover, the architecture used proves to be efficient by optimizing its learning in a few epochs, drastically reducing computational time.

6 Conclusion

Our findings highlight that remote work evokes a range of emotions, predominantly positive ones like joy and happiness, though some individuals express reluctance and dissatisfaction.

LSTM-based recurrent neural networks performed best in emotion classification, emphasizing the importance of balancing performance across emotions to capture distinct patterns effectively. Differentiating similar emotions, such as anger and fear, remains a challenge due to their linguistic similarities. Effective preprocessing—denoising, tokenization, and normalization—proved critical for enhancing model performance. Reddit emerged as a valuable data source for analyzing emotional patterns in diverse communities.

However, emotion detection still faces limitations, especially with nuances like irony and sarcasm. While more advanced state-of-the-art models could address these issues, their significant computational and resource costs prevented us from exploring them in this study. Future work should consider these models to better capture semantic and contextual nuances in language.

References

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Boe and Payne, 2023] Boe, B. and Payne, J. e. a. (2023). Praw: The python reddit api wrapper. Version 7.7.1, available at https://praw.readthedocs.io/.

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras. `https://keras.io`.

[Ekman and Friesen, 1971] Ekman, P. and Friesen, W. V. (1971). Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, 17:124–129.

[Felstead and Henseke, 2017] Felstead, A. and Henseke, G. (2017). Assessing the growth of remote working and its consequences for effort, well-being and work-life balance. *New Technology, Work and Employment*, 32(3):195–212.

[Hochreiter, 1998] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116.

[Honnibal and Montani, 2017] Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

[Machová et al., 2023] Machová, K., Szabóova, M., Paralič, J., and Mičko, J. (2023). Detection of emotion by text analysis using machine learning. *Frontiers in Psychology*, 14:1190326.

[Mohammad and Turney, 2010] Mohammad, S. and Turney, P. (2010). Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34, Los Angeles, CA. Association for Computational Linguistics.

[Oakman et al., 2020] Oakman, J., Kinsman, N., Stuckey, R., Graham, M., and Weale, V. (2020). A rapid review of mental and physical health effects of working at home: how do we optimise health? *BMC Public Health*, 20:1825.

[Parrott, 2001] Parrott, W. G., editor (2001). *Emotions in Social Psychology: Essential Readings*. Key readings in social psychology. Psychology Press, illustrated edition.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Raffel et al., 2023] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. pages 1135–1144.

[Wagner, 2010] Wagner, W. (2010). Steven bird, ewan klein and edward loper: Natural language processing with python, analyzing text with the natural language toolkit. *Language Resources and Evaluation*, 44:421–424.

[Wes McKinney, 2010] Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.