**Depth discussion of word vector selection**

*Consideration of training data*

The availability of emoji vectors is sparse, however a majority train using short form online content such as tweets or Instagram captions. This methodology may have a significant limitation in this context; incongruence in sentiment between emoji and the text may be hard to identify in sarcastic content. Where this is already accounted for within the pragmatic information of the vector itself this incongruence may be less visible. It is unlikely that this would omit incongruence totally as there are no emojis which are used universally to convey sarcasm however emojis which are most frequently used in sarcastic content would be disproportionately impacted. The emoji2vec vectors were trained using official emoji descriptions, which largely omits any noise introduced by emojis being used in contexts contrary to their literal sentiment. However, these descriptions are highly literal and may not capture the nuances of certain use-cases. In the context of use to extract emotion data, this is not necessarily important, however vectors will likely require fine-tuning to provide more useful contextual information when deployed in a sarcasm detection model.

The Google News Word2Vec vectors, trained on news articles are unlikely to be impacted to a significant extent by sarcasm thus by similar logic incongruence is likely to be more evident using this selection. The developers of emoji2vec describe this as the most appropriate set of word vectors to be used alongside their vectors and thus this is a logical choice for use in the regression task. However, there are similar limitations with regards to their use in later sarcasm detection for twitter content given the significant differences in vernacular used between the two sets of text.

*Word vector bias*

Word vectors by nature, carry the biases of the dataset for which they were trained. The impacts of this in the case of the emoji vectors are likely to be minimal given the descriptions which they are trained are highly objective, with no identified use of vocabulary that may introduce bias such as adjectives or adverbs. The converse is the case however in the case of the Google News vectors which have been shown to contain significant bias in terms of gender, socioeconomic status, and race (Bolukbasi et al., 2016). This bias is important to retain within the data as it serves to add context, which is highly valuable to the detection of sarcasm, and sentiment more broadly. However, consideration must be given to whether biases within the media are also representative of that which would be found in the considered tweets. A comparative study relevant to this use-case is a potential area of expansion of this work in future, however the work of Curto et al., 2022 comparing bias in Google News Word2Vec and Twitter GloVe Vectors can provide an indication of the impacts of bias in the word vectors. The work found that the variance in bias was minimal for many topics however the Google News vector displayed greater bias based on socio-economic status and the GloVe vectors displayed greater skew towards negative sentiment in the context of discrimination. This will be a consideration while carrying out hyperparameter tuning.

**Word Vector Optimisation Process**

*Model Architecture*

*Optimiser selection*

Given the complex nature in the relationships between basic emotions, and context playing a key role in pragmatics captured in such vectors, determining a true optimum presents challenges both in defining loss functions and optimiser selection. Two proposed loss functions outlined in section X, aim to exploit known relationships between vocabulary which are postulated to increase sentiment awareness. The complexity of the problem set and high dimensionality of the vectors imply a loss landscape containing many local minima which may hinder optimisation where a function which cannot escape local minima to locate a global minimum is selected. Two optimisation functions were assessed for this task which are robust in escaping local minima:

*Stochastic Gradient Descent:* Gradient descent approaches operate by means of an iterative descent down a slope to locate a minimum for the loss function, defined as a slope equal to zero. To avoid convergence at local minima, the scale of each adjustment at each step is defined as:

Repeated until convergence

Where such a method reduces the step size as the slope approaches zero. Considering this function in the context of a three-dimensional plane, it is evident that there is no mechanism which avoids convergence within a local minimum or saddle point. Hence, an adjustment of this function which leverages a single randomly sampled loss gradient in each step is more appropriate:

Repeated until convergence

This adjustment avoids convergence at saddle points or local minima as the random sample may point away from a local minimum as it may not lie around this particular minimum in the loss contour, allowing the model to escape these points, where the summation of all results would not.

*Adam Optimisation:* Adam is a more sophisticated alternative to the stochastic gradient descent model, which introduces a variable learning rate during training. This method adapts learning rates ()using an exponentially decaying average of past squared gradients and implements an exponentially decaying average of past gradients to update vector direction ():

Where and denote Hadamard (element-wise) product and division respectively and is a smoothing term to prevent division by zero. Given , the update direction has momentum, which pushes the loss away from local minima to locate the global. The adaptive learning rate, is scaled by such that larger gradients result in smaller learning rates. The consideration the two moving averages of the gradients smooths noise, which is likely to feature prominently in the data under consideration. Such smoothing is particularly effective around saddle points, where gradients approach zero in many dimensions.

*Model Tuning*

To select the optimal model for the task, hyperparameter tuning was carried out in both cases to monitor loss over time and best outcome for the loss functions. Hyperparameters assessed in each case were as follows:

*Table X* Summary of hyperparameters assessed.

|  |  |  |
| --- | --- | --- |
| **Model** | **Hyperparameter** | **Values Considered** |
| Stochastic Gradient Descent | Learning rate | 0.00001, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7 |
| Adam | Learning rate | 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7 |
| Batch sizes | 32, 64, 128 |
| Betas | (0.9, 0.999), (0.8, 0.9), (0.7, 0.8) |
| Use regularisations | True, False |

A grid search model was employed to iterate through hyperparameters as the task was deemed not to have excessive computational cost to necessitate skipping permutations of hyperparameters. Relatively large learning rates were assessed to explore the solution space more comprehensively and increase convergence rate. Generally larger learning rates are undesirable due to their unstable gradients and tendency to overshoot optimal points, however given the adaptive learning rate in the Adam model, this effect was offset and smooth descent to optimal points were observed. A larger range considering smaller learning rates was considered for the gradient descent given the constant learning rate. The Adam optimiser additionally considered varying and which control moving average and second moment decay rate respectively. Varying aims to find balance between slow adaptation to mitigate the impacts of noise (smaller value) and faster adaptation which aids the optimiser where large gradients are present (larger value). Similarly, is varied to consider the outcomes where outlier values have reduced impact (smaller value) and greater response to changes in magnitude (larger value). Batch size is considered for the purposes of considering the balance between mitigating noise and reaching the true minimum point rather than navigating around it. The approach to batch size ideally would additionally consider over and underfitting more directly, however in this case fit was assessed through the consideration of the data at varying degrees of loss against human-annotated data.

The Adam optimiser yielded better results with smooth loss curves, contrasting the gradient descent model, indicating that the assumption of complexity in the loss landscape was accurate. The Adam model enabled the initial use of a greater learning rate to navigate this, with mechanisms to escape local minima which was not possible in the gradient descent model due to its constant learning rate.

*Normalisation*

To ensure training would not be prematurely halted due to the direction of vector shifts, and reduce impact vector length related noise, after each iteration vectors were normalised to unit length by dividing each vector by its Euclidean norm. The result is each step moves vectors across a spherical plane of radius 1, rather than leaving the movement plane undefined. This approach preserves cosine similarity values which are not impacted by magnitude, however when considering emotions, it is possible that magnitude may play a role in their identity; for example, it intuitive that *mad* indicates a less intense emotion than *furious* which should be reflected in their respective vector magnitudes. This information is lost using this approach. However, it is unclear the degree to which this information may be of value in the context of the comparisons in question.

\*\*\*Add in 3D surface plots showing the process and discuss\*\*\*

**Model development for basic theory regression**

The overall goal of the model development process was to identify the best possible model for the regression task with regards to its accuracy and robustness. This section discusses in depth the process outlined in section X.

*Data split:*

The data available was initially split into two subsets of train and test data at a ratio of 80:20. This represents a high-typical split for similar use cases, which was deemed appropriate given the small quantity of data available as it retains as much data as possible for the training set, while ensuring the test subset is of sufficient size to give reliable indications of the models’ performance. The purpose of the test data is to withhold some data from the training process to understand how the model performs on unseen data. Within the training process, a secondary data split is performed during cross-validation. These processes were implemented to ensure that recorded performance is attributable to the models’ predictive capabilities, rather than the selection of train and test data for a particular instance.

*Models Assessed:*

The model selection process was created with the purpose of identifying the most optimised and robust model possible for the regression task. Given the limitations in data volume a focus on models with more simplistic architectures was appropriate. The following models were assessed:

*Linear Regression:* Models the relationship between a dependent variable and one or more independent variables by fitting the parameters to a linear equation. This model is simplistic therefore performs well on smaller volumes of training data. The Pearson’s correlation metrics evaluated previously show moderate to high-moderate correlation between parameters and the target variables which indicates that this model may be appropriate for the task.

*Random Forest:* Combines multiple decision tree regressors to predict the target variable. This model is similarly simplistic thus appropriate for the small dataset. If generalisation is a challenge, the aggregation of predictions from multiple decision trees can mitigate the impacts of overfitting. Overfitting has been identified as a potential problem given the predictions not displaying total correlation due to the limitations associated with the vector information and approximation methodologies highlighted previously.

*XGBoost Regression:* An adaptation of the Random Forest regressor which constructs an ensemble of decision trees sequentially, where each tree is modified to correct errors in its predecessor. Predictions are weighted to optimise the overall loss function which enables the capture of more complex patterns than Random Forest, however in this case the increased complexity may be of detriment due to the data volume.

*Support Vector Regression:* An adaptation of linear regression which defines a hyperplane that best fits the data while minimizing points which fall outside of these boundaries. The model reduces overfitting using a margin of error around the plane where the data can reside, and its simplicity enables the implementation with small datasets.

*Gaussian Process Regression:* A model which describes probability distributions over many functions to capture uncertainty and use a probabilistic framework for regression. The model works well for limited sample sized, while still capturing more complex, non-linear relationships and mitigating impacts of noise. Uncertainty metrics may additionally be of value in ensuring the final model is suitably robust.

*K-Nearest Neighbours Regression:* Predicts the value of a target variable by averaging the values of its k closest neighbours. The approach is simplistic and does not rely on large datasets to generate predictions. This is a potentially viable option as throughout the data preparation, an emphasis on generating approximations where similar emotions have similar vectors was implemented, which should return similarity scores which align to similar alternatives. This may not be sufficient given the correlations achieved previously, which may indicate that there is insufficient cohesion between similar emojis to obtain accurate results.

To ensure a comprehensive evaluation of models for the task, several neural networks were assessed for the first univariate regression. While these models generally require larger volumes of training data to learn patterns, they are also capable of modelling more complex patterns. The assessment was carried out to understand if this trade off yielded a favourable result in this case. The models assessed were as follows:

*Feedforward Neural Network:* A model which learns patterns by passing data through sequential layers, applying weighted transformations and activation functions.

*Convolutional Neural Network:* A model which uses convolutional layers to automatically extract relevant features from the input followed by fully connected layers to map the features to the output. It is more commonly used in image processing; however similar implementations have been identified in literature.

*Radial Basis Function Neural Network:* A model which uses a radial basis function as the activation function in its hidden layer. They excel at approximation and interpolation tasks, making them highly suitable for data which contains complex relationships. Additionally, they are effective using data which is unevenly distributed which may improve outcomes given the non-parametric nature of parameters available.

The neural network models are capable of modelling more complex patterns than the previous models, however with increased complexity there is a necessity for a greater volume of data, which is unavailable for this task.

*Target variable format selection:*

The target variables were previously identified to not have a normal distribution and thus alternatives were generated which adjust the values to have a normal distribution. Both these parameters in addition the original labels were assessed to obtain the most optimal outcomes.

The task was identified to have potential for both univariate and multivariate regression. Both options were assessed to ensure comprehensive coverage of consideration to possible optimal models.

*Model Optimisation:*

The optimisation process for the models had two goals. To determine the parameters which generate the most accurate predictions and was robust to altering the data. Hyperparameter tuning was deployed to determine the best possible outcome performance each model was capable of. Cross-validation was implemented to evaluate how robust the performance was.

*Hyperparameter tuning:*

The hyperparameter grids for each respective model were generated with the goal of assessing the impact of a broad range of hyperparameters on the outcomes of the models. Such hyperparameters were identified in the relevant documentation for each model. The values for each hyperparameter in each case aim to cover a broad range of possible options, spanning a range that covers typical values found in similar implementations in literature, adding a buffer above and below the range for comprehensive assessment. Where the hyperparameter has categorical options, these were selected based upon their potential suitability following assessment against the problem set and data available in cases where the number of options was too large for use.

Given the small data volume, the computational cost of the training processes is reasonably low, even where many hyperparameters are present in the grid. For this reason, the grid search method for tuning was deployed to provide the most comprehensive assessment of the selected hyperparameters, which considers every combination of hyperparameters within the defined grid space.

*Cross-Validation:*

To ensure the model performance is not dependent on the specific combination of train and validation data used in a single instance, a cross validation model was implemented to ensure the optimal identified outcome was robust when predicting unseen data. A k-fold method was used which splits the data into five components and combines them to form k iterations of train and validation data. Five folds was the k-value deemed appropriate as this represents an 80:20 split of training and validation data, which was selected per the logic of the split prior to the training stage.

*Dimensionality Reduction:*

The selection of three sets of prediction parameters was implemented with the goal of each subset being able to mitigate the impacts of their respective limitations. However, this strategy operates upon the assumption that the error is inconsistent across each set of approximations, which may not be the case and results in a large quantity of features being introduced during training. This strategy has a significant potential for underperformance without dimensionality reduction due to overfitting, multicollinearity and increasing complexity with a large quantity of features. Several subsets of data were generated by splitting the data into subsets for each approximation method in addition to selectively excluding features with low correlation to the target parameter, per the Pearson’s correlations determined previously.

*Performance Evaluation:*

The models were assessed in terms of their performance based on three metrics which work together to provide a broad picture of the performance:

*Mean absolute error:* Determines the mean absolute difference between predicted and actual values. This metric is selected to provide an easily interpretable metric to understand the error in the predicted values.

*Mean squared error:* Quantifies the average of the squared differences between predicted and actual values, which provides an indication of the overall magnitude of prediction errors with larger penalties applied to greater errors. Given the mean absolute error provides a mean value which does not provide much information regarding the distribution of the error across each individual prediction, the additional information provided by the mean squared error is of value to supplement this limitation.

*R2 score:* Measures the proportion of variance in the target variable which can be explained by the input parameters and can be considered a measure of the ‘goodness of fit’ of the model. This metric was included to provide a more comprehensive understanding of the models’ performance, such as the models ability to generalise which is also essential to understand when considering model performance.

**Selection of pre-trained models for dimensional emotion theory parameter regression tasks**

The purpose of this section is to narrow the scope of potential neural network architectures for assessment for a regression task to generate dimensional emotional theory parameters. This task has been broken down into two phases:

*Phase 1:* Identification of architectures commonly used in relevant literature and critical evaluation of their features with regards to the specific task.

*Phase 2:* Using the most suitable identified architecture from phase 1, assess several pre-trained models which fall into this category for the regression task.

*Evaluation of common architectures in literature for sentiment regression task*

Sentiment is fundamentally linked to context and thus models which successfully capture sentiment are likely to contain characteristics which enable the capture of long-term dependencies. Neural network architectures are widely utilised for such purposes. Their suitability is largely attributable to aspects of their architecture such as their:

*Ability to handle sequential data*: As language is sequential in nature, sentiment is often linked to word order. As neural networks handle data sequentially, their outcomes are often improved compared to more simplistic models.

*Robustness against noise:* Language is inherently noisy and variable due to variance in vernacular across a population in addition to relatively frequent errors in spelling or grammar. Such an effect is known to be more prominent in online content, where such features are sometimes used to add nuance to pragmatics. In the context of a problem set where there is potential for such features to add insight, this feature of such architectures may possess limitations if this information cannot be extracted in another manner.

Several models which would fall under this category are prominently found throughout literature evaluating sentiment analysis methodologies and specifically in the domain of sarcasm detection due to these features. The three primary models identified were as follows:

LSTM: The LSTM mechanism of selective memory is made up of a cell state, hidden state, and gates. The NN models sequential data by propagating over time through the connection of sequential events using the hidden state. This component captures dependencies by considering both the previous step and current output:

However, a feature of such a mechanism result in all previous steps being considered in the current step when implemented in isolation due to the chain rule:

Because:

With the ultimate result being either vanishing or exploding gradients and thus limitations on the abilities to capture long term dependencies in isolation. The cell state mitigates this effect through the filtering of less relevant information from the memory through the forget gate. This information, in addition to the weights from the input gate enable the model to learn which time steps contain important information, resulting in weights for each time step being represented in proportion to their understood importance to the model. Literature evaluating this architecture is not consistent with regards to its assessment of the efficacy of such a mechanism; with some studies citing the model as effective to capture long-term dependencies, and others postulating that the mechanism may ‘dilute’ important information over time. Similar contrasting observations are found in the sentiment across literature with regards to their robustness against noise, with the former asserting that noise information is filtered efficiently during training and the latter arguing the converse and observing amplification of noise. In this context, what is traditionally regarded as noise may provide pragmatic cues with respect to potential sarcasm as discussed above, these cues may be indicative that context of the problem set plays a part in the efficacy of the architecture to model valid patterns in the data, which explains contrasting observations within literature. No works could be identified which assessed this hypothesis, however this may be an area for future research.

GRU: GRU models address vanishing and exploding gradients using a more simplistic mechanism than LSTM models. These models omit the cell state and regulate memory using gates. These architectures utilise an update gate to dictate the information which is retained from the previous step in series with a reset gate which dictates the information which should be eliminated. Compared to LSTMs for natural language processing tasks, due to their more simplistic architecture GRU models seem to perform better where shorter data sequences are used, possibly providing greater potential in the context of the short form content as is used for the problem set in question.

Present state-of-the-art for sentiment analysis also includes significant volume of models which contain an alternative mechanism for memory. Transformer-based models leverage self-attention mechanisms to capture dependencies of long and short ranges.

These models are based upon encoder-decoder architectures, which are capable of processing data in parallel due to their attention mechanism which avoids processing data in parallel in favour of processing the sequences as a whole. The encoder consists of several layers, each containing two sublayers. The first sublayer generates self-attention and the second consists of a fully connected feed-forward network with two linear transformations and Rectified Linear Units activation:

Where each layer uses its own weights and bias parameters. Given there is no inclusion of recurrence, there is no embedded manner to consider the relative position of words. To address this, positional encodings are added to the word embeddings. The decoder consists of several layers, which are each composed of three sublayers: the first decodes the previous input to extract positional information and apply attention. The attention in decoders is distinguished from that found in the encoder cells as they do not consider all words, but rather only words which have occurred before the current. The second layer contains a self-attention mechanism which receives information from the previous sublayers of decoders and the encoders output keys and values. The final decoder sublayer consists of a fully connected feed-forward neural network, like that of the second sublayer in the encoder cells.

The attention layers operate by passing each word in the sequence through the embedding and positional encoding layers to generate their respective vectors. The result is passed into the encoder where it is first processed by the attention module. The sequence is passed through three separate layers which each produce a matrix. These layers consist of query which defines the word for which the attention is to be calculated and key and value are compared to the query with regards to their relevance. These transformations are trainable operations which are adjusted to produce the desired output predictions over the course of training, quantified by the attention score, defined as the dot product between the query matrix and a transpose of the key matrix:

An intermediate matrix is produced, consisting of a multiplication between all combinations of the words in the respective matrices. A second dot product calculation is performed between the intermediate matrix and the value matrix to produce the attention score.

A diagram of a number

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A diagram of a graph

Description automatically generated

Given such a mechanism which learns based on similarities and differences in the input words, the architecture is aligned with the previous methodology for word vector transformation which yielded improved results compared to the original word vectors used in the basic theory regression task. Additionally, the lack of necessity for labelled data is advantageous adaptations of the strategy implemented for the basic theory regression could not yield acceptable levels of accuracy. The key characteristics of the training data which may have contributed to this result may be the small size of the training set, and the potential for complexity in the relationships between the emoji and labels, which could not be captured by less complex algorithms, more appropriate for the available training data. While traditional neural networks necessitate large, labelled datasets, the converse is true for transformer-based models which learn based on patterns between elements in the dataset, eliminating the need for such a resource intensive annotation process.

Each identified option presents advantages and disadvantages with regards to their architecture, and evaluation of each option may be the best approach to determining the best performance in this use case. However, considering the identified neural network options which necessitate annotated data, no models were identified which were trained using emoji and thus these options were omitted from consideration. Transformer-based models for dimensional theory sentiment analysis trained using emoji were identified, falling into the BERT category. Several iterations of BERT models tuned for various contexts to achieve outputs of dimensional-based sentiment scores were evaluated for correlation to the human-annotated dataset using Pearson’s correlation. The optimal model identified was the latest Twitter RoBERTa Base Sentiment which achieved a Pearson’s correlation of 0.83 to the human annotated sentiment scores. Fine-tuning the model using the human-annotated data did not result in improvement to this score, likely due to the small quantity of data available for the purpose therefore the model was deployed for the regression task without optimisation.

*Evaluation of methodology*

While the results obtained yielded good correlation to the human annotated data, the input of a single emoji in each case to obtain the results does not necessarily align with the intended use for the model; the attention mechanisms cannot provide contextual information from surrounding information and thus outputs may be limited to some degree by ambiguity. Alternative methodologies would likely involve the input of emoji-containing strings of text, however in the context of the problem set this also has limitations; the sentiment may be skewed by the content of the text. This concept also applies to some extent even where the emoji is inputted in isolation due to the training data. While the approach has limitations, the solution would involve the obtaining of a larger training dataset with annotations. Such a task is outside the scope of this work however may be an avenue for future consideration.

**Optimisation Process for Sentiment-Aware Vector Space Modification Model**

**Architectures of Considered Emoji Sentiment Prediction Models**

**Survey Results**