Image from deepAI image generator based on prompt: “Eye-movement based classification of unknown words with LSTMs and CNNs”

Word Familiarity Classification based on Eye-Movements using Neural Networks.

William LaCroix
wila00001@stud.uni-saarland.de
Matrikelnummer: 7038732
Saarland University
Winter Semester 2023-2024

1. Introduction

This project is a methodological replication study which presents a follow-up to Ryzhova et al. 2023 - *Word Familiarity Classification from a Single Trial Based on Eye-Movements. A Study in German and English.* In the original paper, the authors present the results of a logistic regression model trained on subsets of their experimental data set, depending on the task (more detail in section 3 - Methodology / Modelling). As we all know, neural networks are just better than statistical models[1], but in order to quantify this difference, we will need to develop and train our own models for prediction. The research question at hand is fairly simple: how much more accurate and precise are neural models when compared with logistic regression at predicting unknown words from eye-movement features? The hypothesis is that neural networks, specifically Recurrent Neural Networks (RNN) like Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) will be able to learn patterns from the data that are not extractable by standard regression techniques. Due to their ability to capture patterns at a distance, the hope is that RNNs can outperform the relatively simple feedforward networks, which in turn outperform traditional regression techniques.

I will give a brief overview and description of the dataset, including available and trained-on input features, describe the modeling approach used to train the classifiers, and explain the various tasks to be performed and tested. Following this I will present the results of my experiments and discuss the model performance in comparison with that of the original paper, as well as the limitations faced when building these models.

2. Methodology / Modelling

2.1. Sample Description

The dataset consists of 32 participants, each reading 28 sample sentences while their eye movements were measured. These 32x28 samples were then split into data points per word, resulting in a total of 11,176 entries, after discarding a number of entries for various reasons that are beyond the scope of this paper. The experimental manipulation was the use of a critical word in a subset of the 28 sentences. 3 of those

[1] Citation needed

critical sentences had the critical word in the 'pseudo' condition – that is, a made up pseudoword – while an additional 9 sentences had the critical word in the 'control' condition; an unusual word that is nevertheless real. The remainder of the non-critical sentences contained only filler words. Of the original 11,176 data points, 343 were critical words: 85 pseudowords and 258 control words. While these filler words were the foundation for the statistical model presented in the original paper, including the filler words in the training data would only exacerbate the class imbalance of 3 positive examples to the 9 negative examples for training a neural model.

2.2. Input features

The original study included subject-specific eye-movement measures (first-duration fixation, first-pass duration, dwell time, number of fixations, and number of regressions out), as well as subject-independent word features (word length, log word frequency, log trial order) as predictors. It is often important to reduce input features for regression models, to prevent overfitting and increase interpretability, but since neural models are black boxes by nature, it seems reasonable to take an "kitchen-sink" approach to model input; including as many input features as are available in the original dataset. However, for this model to make its predictions without data leakage, it is important to choose only those input features related to eye-movement measurements, and exclude metaknowledge about the word positions and linguistic features (frequency, etc.) Thus the final input features used to train the models are as follow:

> ['fixation_duration', 'duration_firstpass', 'duration_firstfixation', 'fix_count', 'avg_pupil', 'IA_REGRESSION_IN_COUNT', 'IA_REGRESSION_OUT_COUNT', 'saccade_length', 'saccade_duration', 'go_past_time']

While the following features were left out of the final model:

> ['composite', 'LF', 'HF', 'IA_ID', 'trial', 'list', 'IA_LABEL', 'wordlength', 'is_critical', 'is_spill1', 'is_spill2', 'is_spill3', 'filler', 'function_word', 'other_filler', 'condition', 'sentence_condition', 'recording_session_lable', 'item']

2.3. Modeling approach

The objective of these models is to classify the critical words in the dataset as either {0: control | 1: pseudo}, based on eye-movement measures alone. To that end, two approaches may be taken at different resolutions: sentence-level and word-level prediction.

Word-level prediction: use the input features to classify all words as **known|unknown**. This has the advantage of expanding the training data size from *N* sentences, to *N x L* words, and can be viewed similarly to a machine translation task with a very limited target language vocabulary, or named entity recognition task with only 2 labels {known, unknown}. The disadvantage to this approach is, as mentioned before, the highly imbalanced class distribution. By expanding the task to predicting all words, the dataset includes all 11,176 data points, but the positive class label amounts to only 3.07% of the dataset. This imbalance means our model would achieve nearly 97% accuracy by simply predicting "not-pseudo" in all cases. Problem solved, accuracy improved. Except that a model with only negative predictions results in an F1 score of 0. Not good.

Sentence-level prediction: use the input features to predict whether the target word is **control|pseudo**, giving a probability distribution over {0,1} as output per sentence. By examining only critical words/sentences, this method may suffer from the relatively small size of the dataset: by limiting the task to critical words only, we reduce the dataset size from 11176 to only 343 data points, at a ratio of 1 positive to 3 negative.

For the reasons illustrated above, I chose to focus on critical word classification, without expanding the model to include filler words, though this did not fully solve the class imbalance issue. A weighted binary cross entropy loss was used to calculate the loss for the weight updates, but I found that the model performance was highest with a custom class weight scaling factor I termed *alpha*, since *beta_1* and *beta_2* were already named hyperparameters in the optimizer; in this case AdamW. To find optimal values for these and hyperparameters, a brute force beam search method was used to test hundreds of model configurations, and the best combinations of hyperparameters were used in the final model. Those parameters were:

- Alpha: 0.21 (note that alpha is used in place of class weights to dynamically adjust class label balance. More on that in the section on cross-validation)
- Beta_1 / beta_2: 0.999
- Learning rate: 0.001
- Hidden layer size: 500
- Number of hidden layers: 6
- Batch size: 16

These values were shared across the majority of tasks, with a few exceptions: For task 1, class POS weight of 0.65 was used. For task 2, alpha was set to 0.19. For task 3, class POS weight was set to 0.8, beta_2 was set to 0.99, and a small dropout of 0.01 was added for regularization. Of note is that most models did not benefit from including dropout or batch normalization, though performance did increase when the input data was normalized during preprocessing. Finally, a weight decay of 0.01 was used in the model optimizer.

2.4. Model architecture and training

As discussed above, the final model used a series of 6 linear layers, with LeakyRelu as the activation function, and (optionally) a dropout layer in between. Figure 1 below is a largely unnecessary illustration of the final MLP model architecture. A batch of eye-movement features corresponding to the critical words is used as the model input, which gets expanded to the hidden layer size of 500 nodes, and transformed 5 more times at that hidden layer size before being projected down to the a single value, which is passed through a sigmoid function for generating predictions.

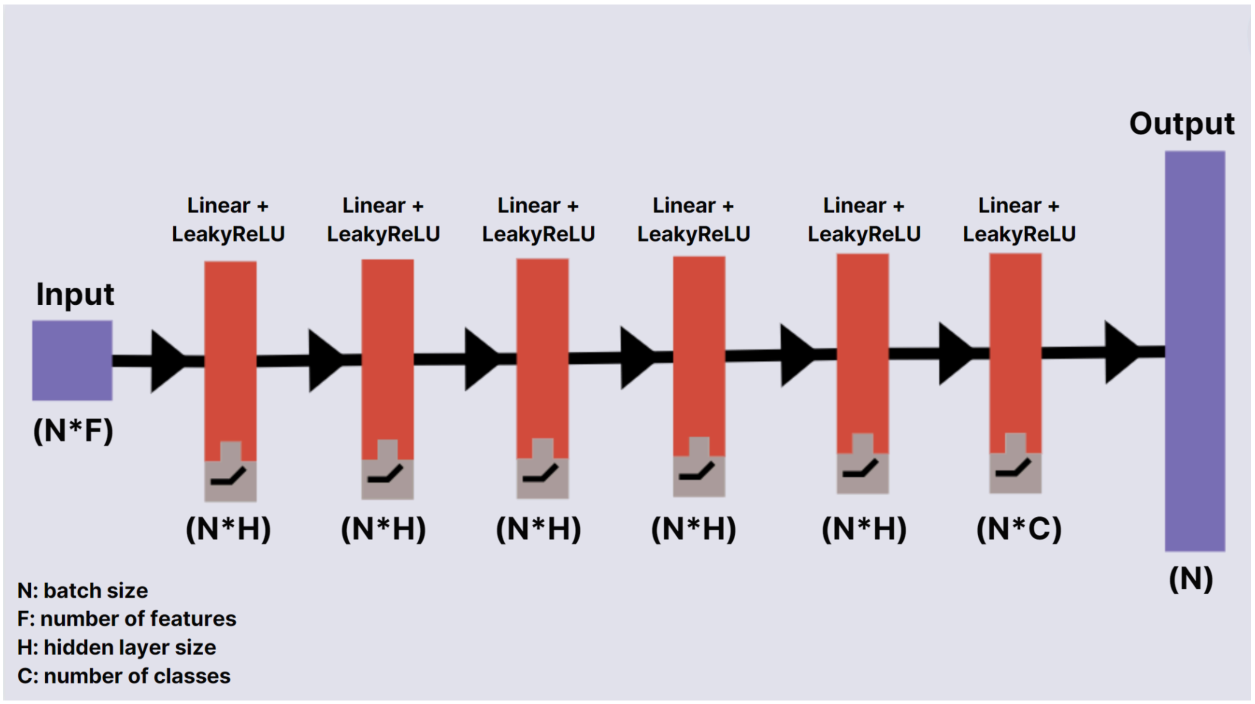

Figure 1: Multi-Layer Perceptron model architecture.

Image generated from the Ennui neural network visualization tool at https://math.mit.edu/ennui/

Because neither dropout nor batch normalization were used in the majority of models, early stopping was employed as a regularization technique to prevent overfitting. A maximum patience of 10 epochs was set, allowing the model to wander a little before converging. Though the majority of models in LOOCV setting converged within the first training epoch, there were a number of splits whose convergence took upwards of 30 epochs of training. This is likely due to individual differences in subjects or items during testing, and potentially explains the task 2 and 3 (explained below) results.

2.5. Cross-validation and task robustness

Due to the nature of the dataset, both in terms of overall size, as well as the repeated measures for each subject and item, one major concern is model robustness and generalizability. To test this, the original authors tested their models in three distinct task settings:

1) Seen items, seen subjects. This is the default data split, covering all study participants and critical items. The dataset is shuffled and split into training, validation, and test sets. In addition to 10-fold cross validation, the small size of the dataset made leave-one-out cross validation a viable experiment to run, as a given model takes only a handful of seconds to converge when training on GPU. According to the Machine Learning Bible, An Introduction to Statistical Learning with Python (Hastie et al. 2023), k-fold cross validation is a biased estimator, since the left out data introduces additional model variance, if the individual folds are not themselves similarly representative of the whole dataset. On the other hand, LOOCV is considered to be an unbiased estimator, since model training takes account of all but one data point in forming its representations. In most cases, LOOCV is prohibitively expensive in terms of computation required, hence k-fold CV serving as an approximation, but as mentioned above, this is not an issue in the current study. Therefore, I perform cross validation on task 1 in two forms: 10-fold CV (80:10:10 train:eval:test split) for comparability against the initial study, and LOOCV with one data point held out for both validation and testing, (n-2:1:1 train:eval:test split) to get the most representative model performance possible.
2) The second testing scenario is left-out subjects, where all data points from a given subject (or 2 subjects, as in the original study) are left out, and the rest are used

for training (~90:~10:1 subject OR ~90:~10:2 subjects) This will ideally test the models robustness to individual variation, since strong performance on unseen subjects would indicate that the model's predictions generalize well across individuals.

3) The third and final testing scenario is left-out items, where all data points from a given critical sentence are left out for testing, and the remainder are split into training and validation sets (~90:~10:1 item). As opposed to left-out subjects, left-out items testing tests the model's ability to predict previously unseen novel words and ability to overcome the variance inherent in individual vocabulary items.

3. Results

Table 2 below shows the results of testing across these 3 different tasks, including various levels of cross validation. Somewhat shockingly, in task 1, the MLP model achieves perfect test predictions using LOOCV. It seems that having access to a variety of examples per subject and per item is enough to allow the model to predict the singular left-out data point. Performance decreases slightly for the MLP 10-fold CV, but still outperforms the regression model from the original study. Coming in last for task 1 was the bidirectional LSTM model, topping out at an F1 score just below 0.8. The LSTM's poor performance, coupled with the exceptional performance of the MLP model led me to abandon that line of investigation relatively early on. Future work may extend this study to include a wider variety of more thoroughly tested LSTM and CNN models, but since they weren't going to improve on *perfect*, it seemed an unnecessary extension.

Performance on task 2: left-out subjects drops below that of LOOCV for 1 left-out subject, and below 10-fold CV for 2 left-out subjects. Cross validation was performed at these two granularities in order to: 1) replicate the original study's conditions as closely as possible (in the case of 2 left-out), and 2) take the narrowest possible view of the training data in the case of 1 left-out subject, in the same vein as task 3 using 1 left-out item. Performance here is very impressive, since it indicates the model's ability to generalize beyond the subjects encountered in training. In both task 1 and task 2, it was false positives which contributed most to model error. The fact that model error is not evenly split between false positives and false negatives likely goes back to the issue of class label imbalance not being perfectly resolved by weighted cross entropy loss or alpha weight scaling.

On the other hand, performance on task 3: left-out items showed a slight favor toward false negatives (3 false negatives to 2 false positives), while still outperforming the regression model. Confusion matrices for each of these runs are shown below, in table 3.

Task	Model	Accuracy	F1
1	**MLP LOOCV**	**1.0000**	**1.0000**
	MLP 10-fold CV	0.9825	0.9643
	Regression 10-fold CV	*0.9531*	*0.9032*
	LSTM 10-fold CV	0.9009	0.7930
2	**MLP leave-1-out**	**0.9854**	**0.9701**
	MLP leave-2-out	0.9708	0.9405
	Regression leave-2-out	*0.9479*	*0.8913*
3	**MLP leave-1-out**	**0.9708**	**0.9419**
	Regression leave-1-out	*0.9531*	*0.9032*

Table 2: Accuracy and F1 scores by model and task

Task 1	**LOOCV**	*Predicted*	
		Positive	Negative
Actual	Positive	**285**	**0**
	Negative	**0**	**85**

Task 1	**10-fold CV**	*Predicted*	
		Positive	Negative
Actual	Positive	**256**	**4**
	Negative	**2**	**81**

Task 3	**1 left-out item**	*Predicted*	
		Positive	Negative
Actual	Positive	**255**	**2**
	Negative	**3**	**83**

Task 2	**1 left-out subject**	*Predicted*	
		Positive	Negative
Actual	Positive	**257**	**4**
	Negative	**1**	**81**

Task 2	**2 left-out subjects**	*Predicted*	
		Positive	Negative
Actual	Positive	**254**	**6**
	Negative	**4**	**79**

Table 3: MLP confusion matrices for tasks 1-3

4. Discussion and conclusion

I set out on this quest with one goal in mind: improve upon the baseline established in the original study. My path even seemed clear, since it was obvious to me even before I began that the recurrent nature of LSTM and CNN networks would make them superior to the statistical models, and even to the other neural architectures. I ended up reaching my goal, but it was not via the path I had expected. In all tasks and all variations of cross validation, the neural models outperformed the statistical model developed in the original paper, with LOOCV even achieving a perfect testing F1 score. This is a huge step forward in terms of the predictive capabilities of these neural models when compared against a more traditional statistical model. Even now, the typical limitations of model and dataset size did not serve as a major hurdle for model training and development. Even though my research hypothesis – that RNNs would outperform both statistical and feedforward models – was not supported, the outstanding results achieved by these models is nothing short of amazing. As mentioned earlier, given additional time, I would like to revisit my original hypothesis and develop/extend LSTM and CNN networks to more thoroughly assess whether or not their structure is suited for this task. But until then, suffice it to say the feedforward models definitely are.

Thank you.

Project repository:

https://github.com/WilliamPLaCroix/EyeTracking.git

References

Margarita Ryzhova, Iza Škrjanec, Nina Quach, Alice Chase, Emilia Ellsiepen, and Vera Demberg. 2023. Word Familiarity Classification From a Single Trial Based on Eye-Movements. A Study in German and English . In 2023 Symposium on Eye Tracking Research and Applications (ETRA '23), May 30–June 02, 2023, Tubingen, Germany. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3588015.3590118

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in Python. New York : Springer, 2023.