Volume 14 Issue 1 (2025)

Enhancing Automated Autism Detection with Improved Word Embeddings

Yao Chen[1] and Darnell Granberry[#]

[1]The Quarry Lane School, USA
[#]Advisor

ABSTRACT

Autism Spectrum Disorder (ASD) is a developmental disorder that affects a significant amount of people. Unfortunately, ASD manifests in a large number of ways, meaning that diagnosing ASD is both time-consuming and inaccurate, which results in many children with ASD not being diagnosed until later childhood. One advantage of an early diagnosis is that it allows for early intervention, which typically leads to much better results. We investigated multiple different machine learning models as potential methods of predicting ASD in children from text segments provided by the child's caregiver. Two promising models are multilayer perceptrons and logistic regression. We investigated different hyperparameters for multilayer perceptrons, such as the number of layers and size of hidden layers. We then conclude that text descriptions of a toddler's behavior given by a caregiver are highly accurate for predicting autism when combined with a multilayer perceptron.

Introduction

Autism spectrum disorder (ASD) is a developmental disorder that is commonly associated with difficulties in social communication and interaction, along with repetitive behavior and/or restricted interests. [1] The symptoms of autism vary among individuals and not all symptoms are necessarily present in an individual with autism. Therefore, autism diagnosis is a tedious and time-consuming process with rigorous screening throughout a child's early years. This process often takes many years, so children with autism often are not diagnosed until early childhood. [2] If autism is diagnosed early, then this allows for therapies such as Applied Behavior Analysis to be provided earlier, when it is more effective. Early diagnosis of autism is associated with benefits later in life, such as a lower risk of mental health conditions in adolescence. In this project, we attempted to use natural language data provided by the caregivers of toddlers to help predict whether the toddler has ASD, a supervised learning problem. We investigated using a variety of architectures, such as multiple multi-layer perceptrons, as our machine learning model.

Dataset and Data Preprocessing

As mentioned above, the dataset we chose was the "Text-based Early Autism Spectrum Disorder Detection Dataset for Toddlers" dataset. [3] To generate the dataset, caregivers of toddlers were asked questions describing their toddler's behavior in different categories. Each sample within the data consisted of the text response, along with the category of the behavior that is described, which was not used in the model, and whether the child whose behavior was described has ASD. There were a total of 298 text responses. Of these, 139 described toddlers who were neurotypical (no ASD) and the other 158 described toddlers with ASD.

The data type that models such as MLPs take as input is vector data, which is different from the raw natural language text data found in the dataset. As a result, the text must be converted into vector data in order for the MLP to be able to analyze the data. The first step is to tokenize the data, which is to split each text segment into a list of individual words. The second step is to remove stop words. Stop words are common words that carry little information about the overall text, which can negatively impact the model performance. The third step is to stem the data. Each word in the dataset that isn't a stop word is converted to the base form of the word. Usually, prefixes and suffixes are removed, but the word form may be changed from plural to singular even without removing suffixes. The fourth step we took was to use word-2-vec to convert the base forms of the words to a vector representation. We used a pre-trained word-2-vec model for this step, as described by Mikolov et al. (2013). [4] The model maps each word to a vector. The model was trained on a large corpus of text to determine which words have similar meanings, and assigned each word a vector so that words with similar meanings have similar vector representations. The final step is to take the average of the vector representations of each word within a text segment to get an overall vector representation for the whole text segment. If a word is repeated multiple times, it will be weighted more heavily in the average, which may impact the model output. All information about word order is lost during this process. Although word order information may be valuable, we do not know whether it is or how much so. One alternative, which does take into account word order, is using n-grams, but that has many drawbacks, as explained in the section above.

Methodology

The first step I took was to split the preprocessed data into training data and test data. 20% of the data points were randomly chosen to be the test data and the remaining 80% of the data points were the training data. This process did not take into account the question category or the ASD labels of the data, so the training or testing data could have a different ratio of data points that correspond to different categories or ASD statuses of the data. Since there are so few data points for each question category, using the category labels might lead to unreliable results due to the amount of variance present in a small sample. Because of that, we removed the category labels.

This problem is quite similar to a sentiment analysis task since the objective is to analyze a person's description of a toddler's behavior. In this classification task, a label of 0 corresponds to the data representing the behavior of a neurotypical toddler, and a label of 1 corresponds to the data representing a toddler with ASD.

The next thing I did was to test multiple different models to see which was the most effective when the data was pre-processed using word-2-vec, in contrast to n-grams. Different models operate differently, so they might give different results. To train each model, a loss function is used, which is the sum of the loss values for all data points in the training set. In this case, I am using the Binary Cross-Entropy Loss function, meaning that the loss for each input is the negative log of its probability.

The first model I used was logistic regression, which involves interpolating the logistic value of the probability (i.e. the logit) that a given vector representation falls into the ASD class. One key idea here is that the range of possible probabilities [0,1] is mapped to the set of real numbers $(-\infty,\infty)$ and back through the logistic and sigmoid functions, respectively. A linear function converts the vector values into a single number, which is then mapped to a probability using the sigmoid function. To train this model, the linear function's weights were adjusted so that the loss value was minimized. [5]

The second model I used was a random forest. This model consists of a large number of decision trees, and it passes the input into each of the decision trees within the model. In other words, a random forest is an ensemble of a large number of decision trees. Each decision tree consists of a set of nodes that contain a decision to make. The decision tree starts at the root node, and at each node, a decision is made that determines the next node to go to. Eventually, the decision tree will reach a leaf node, which contains a category label instead of a decision, and it outputs the category of the leaf node. The probability assigned to each category is proportional to the number of decision trees that outputted that category. [6]

The third model I used was a multilayer perceptron. This model consists of multiple layers of neurons, including the input layer, at least one hidden layer, and an output layer. The neurons in the first layer consist of the input values of the vector input. Each neuron on the second layer onwards is connected to each neuron in the first layer. To compute the value of each neuron in the second layer onwards, the values of each neuron in the previous layer is multiplied by the corresponding weight for the neuron pair, and the sum of these values plus another bias value becomes the value. We typically model this as a matrix multiplication and addition. An activation function is then applied to the value of each neuron, which in our case is the rectified linear unit (ReLU), which sets negative values to 0 and leaves positive values unchanged. The final layer has two output neurons, and the probability assigned to each category depends on the values of the two output neurons. [7]

The fourth model I used was an ADA boost classifier. The ADA boost classifier consists of a series of smaller models that were trained one by one. For each model, the training data points that were classified incorrectly by the previous model are weighted more heavily during the training process. The overall model is an ensemble of all the smaller models. The smaller models are weighted based on their accuracy during the training process. [8]

The final model I used was a support vector classification. This model treats the data as a n dimensional vector, where n is the number of data points in the input. During the training process, the model tries to find a n-1 dimensional structure in the space of possible vector values so that the distance between the two categories is maximized. [9]

The next thing I did was change hyperparameters for the MLP, since it seemed to be the most promising model. [10] I changed the number of layers in the MLP and tested 3 layers up to 10 layers, while keeping the number of neurons in each hidden layer (All layers except the first and the last) at 100. Then I changed the number of neurons of each layer and tested the models. The numbers of neurons I tested were the multiples of 20 from 40 to 160. I tested both 3 and 4 hidden layers because too many layers, especially for a small dataset size, might lead to overfitting. To reduce random error, I trained three independent copies of the model for each set of hyperparameters I was testing. I wanted to see which configuration led to the greatest accuracy, so I plotted the accuracy scores for each against the number of layers and the number of neurons in each hidden layer to see the overall pattern.

Results and Discussion

To test each model, the model was used to classify each data point in the test set, and the predicted categories were compared with the actual categories to get the metrics.

The main metrics I used were accuracy, precision, recall, and F1-score. Accuracy is the proportion of data points that the model classified correctly. Precision is, among the data points that were classified as positive (with ASD), the proportion of these data points that are truly positive. Recall is, among the data points that are truly positive, the proportion of these data points that were classified as positive. F1-score is the harmonic mean of the precision and recall scores, or the reciprocal of the average of the reciprocals of the precision and recall scores. As shown in Figure 1, the multiple MLPs and logistic regression appear to perform similarly and best by accuracy and F1 score. [11]

	Accuracy	Precision	Recall	F1-score
Random Forest	0.87	0.90	0.76,	0.83
Multiple MLPs	0.88	0.95	0.76	0.84
Ada Boost	0.78	0.7	0.84	0.76
SVC	0.82	0.75	0.84	0.79

Logistic Regression	0.88	0.91	0.8	0.85

Figure 1. The metrics for the different models. Each row corresponds to a model type and each column corresponds to one of the four metrics.

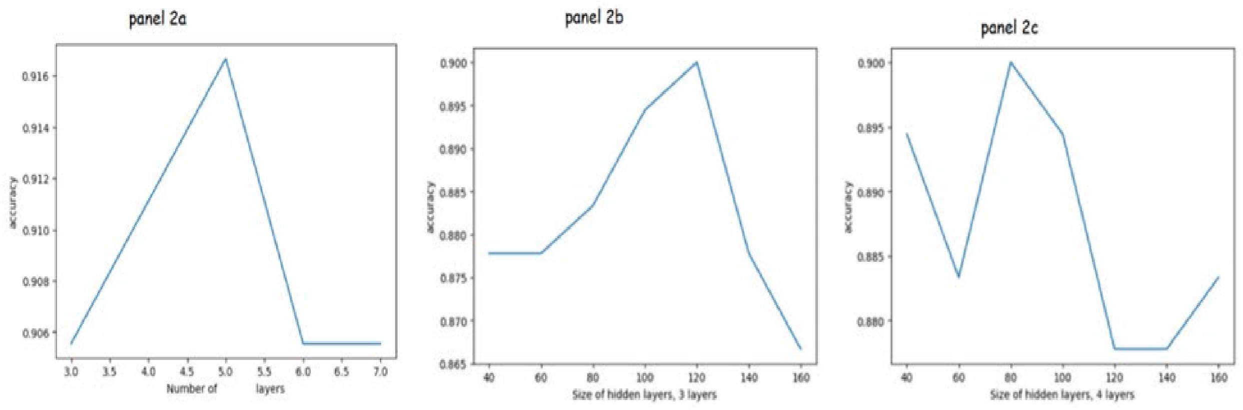

Figure 2. MLP test accuracy versus the hyperparameters. For panel 2a, the horizontal axis is the number of layers. For panels 2b and 2c, the horizontal axis is the number of layers. Each size of the hidden layers was tested three times to reduce random effects; we plot the mean accuracy.

As shown in Figure 2, the MLP accuracy peaked at five layers, but the number of iterations required to train the model also increased. At higher numbers of layers, overfitting might occur, which is when the model is too specific for the values within the dataset and forgets about global patterns, which reduces accuracy.

The second hyperparameter I tuned was the size of the hidden layers. For an MLP with 3 layers, the accuracy peaks at 0.9 at 120 neurons per hidden layer. For a MLP with 4 layers, the accuracy peaks at 0.9 at 80 neurons per hidden layer. The main reason why the accuracy decreases at high hidden layer sizes is due to overfitting. These results are significant because of the high accuracy. While the models described by Chistol et al. (2024) had accuracy scores that peaked at 78%, most models we tested had an accuracy score in the range of 80-90%, and even Ada boost–my worst model–had the same accuracy score as k-Nearest Neighbors, their best model. [2] When changing the hyperparameters, higher accuracy scores were observed, including accuracy scores of at least 90%. This shows that word-2-vec is a much better preprocessing method than n-grams, at least for this small dataset.

Conclusion

Logistic regression and multilayer perceptrons are effective methods of predicting autism of toddlers from text segments given by the caregiver. We took a dataset of text responses of caregivers to questions that describe a child's behavior, and trained different kinds of models on the data, while also changing some hyperparameters. The two most promising models were linear regression and multilayer perceptrons. I believe that my model had a high accuracy because the method of preprocessing, word-2-vec, is able to extract the absolute meaning of the text, which increases the amount of data received by the neural network. In previous research, a lot of data is lost in the preprocessing step because the preprocessing method is unable to account for similarities in the word meanings.

Some next steps would be to preprocess data using BERT, or any other large language model that can convert a whole sentence into a vector representation. This would allow the neural network to access information about the word order within each text segment. In addition, all the responses for a given toddler can be used and converted into a single vector to increase the amount of information given to the machine learning model. Another next step is to increase the dataset size. I can potentially interview more parents and caregivers of toddlers who may or may not have autism. If fully supervised learning is to be used, I would then need to wait multiple years for the toddlers to be diagnosed with autism. On the other hand, semi-supervised learning can be used, where the model can use a limited amount of labeled data along with a larger amount of unlabeled data to categorize data points.

My model has immediate practical implications. For example, parents of toddlers may be asked several questions about their toddler's behavior, and my model can predict if the toddler has autism. The model cannot predict autism with certainty, but the results can still be used for pre-screening, where toddlers who are predicted as likely to have autism may be identified for further screening.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

1. American Psychiatric Association, Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition. American Psychiatric Association, 2013. doi: 10.1176/appi.books.9780890425596.

2. Chistol, Mihaela; Danubianu, Mirela, “Automated Detection of Autism Spectrum Disorder Symptoms using Text Mining and Machine Learning for Early Diagnosis” International Journal of Advanced Computer Science and Applications(IJACSA), 15(2), 2024. http://dx.doi.org/10.14569/IJACSA.2024.0150264

3. Hachemi, Rania; Degha, Houssem Eddine (2024), “TASD-Dataset: Text-based Early Autism Spectrum Disorder Detection Dataset for Toddlers”, Mendeley Data, V2, doi: 10.17632/87s2br3ptb.2

4. Mikolov, Tomas; Chen, Kai; Corrado, Greg; Dean, Jeffrey (16 January 2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781

5. “Logistic Regression in Machine Learning” GeeksforGeeks, 9 May 2017, www.geeksforgeeks.org/understanding-logistic-regression/. Accessed 26 Oct. 2024.

6. “Random Forest Algorithm in Machine Learning” GeeksforGeeks, 22 Feb. 2024, www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/. Accessed 26 Oct. 2024.

7. “Classification Using Sklearn Multi-layer Perceptron” GeeksforGeeks, 11 Oct. 2023, www.geeksforgeeks.org/classification-using-sklearn-multi-layer-perceptron/. Accessed 26 Oct. 2024.

8. “Boosting in Machine Learning | Boosting and AdaBoost” GeeksforGeeks, 3 May 2019, www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/. Accessed 26 Oct. 2024.

9. "Support Vector Machine (SVM) Algorithm" GeeksforGeeks, 20 Jan. 2021, www.geeksforgeeks.org/support-vector-machine-algorithm/. Accessed 26 Oct. 2024.
10. Li Yang, Abdallah Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, Neurocomputing, Volume 415, 2020, Pages 295-316, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2020.07.061.

11. Erickson, Bradley J., and Felipe Kitamura. "Magician's corner: 9. Performance metrics for machine learning models." Radiology: Artificial Intelligence 3.3 (2021): e200126. https://doi.org/10.1148/ryai.2021200126