Spring 5-28-2019

Classifying Challenging Behaviors in Autism Spectrum Disorder with Neural Document Embeddings

Abigail Atchison
Chapman University, atchi102@mail.chapman.edu

Recommended Citation

A. Atchison, "Classifying challenging behaviors in Autism Spectrum Disorder with neural document embeddings," M. S. thesis, Chapman University, Orange, CA, 2019. https://doi.org/10.36837/chapman.000071

Classifying Challenging Behaviors in Autism Spectrum Disorder with Neural Document Embeddings

A Thesis by

Abigail Sandra Atchison

Chapman University

Orange, CA

Schmid College of Science and Technology

Submitted in partial fulfillment of the requirements for the degree of

Master of Science in Computational and Data Sciences

May 2019

The thesis of Abigail Sandra Atchison is approved.

[illegible]

Erik J. Linstead, Ph.D., Co-Chair

[illegible]

Elizabeth Stevens, Ph.D., Co-Chair

[illegible]

Dennis Dixon, Ph.D.

May 2019

ACKNOWLEDGMENTS

I would like to thank my family, my friends, and the Machine Learning and Assistive Technologies (MLAT) lab for supporting me on this journey. More specifically, I would like to thank my advisor and friend Dr. Erik Linstead for seeing my potential and equipping me with the knowledge to pursue it and my brilliant colleagues Adrienne Bergh, Dr. Elizabeth Stevens, and Jordon Ott for constantly inspiring me and serving as a support system throughout this project and my time here at Chapman. The growth I have experienced and the work I've been able to do these past 4 years would not have been possible without any of these individuals and organizations and for that I am forever grateful.

VITA

Abigail Sandra Atchison

EDUCATION

Master of Science in Computational and Data Science **2019**

Chapman University *Orange, CA*

Bachelor of Science in Computer Science **2018**

Chapman University *Orange, CA*

INDUSTRY EXPERIENCE

Software Engineer Intern **June 2018-August 2018**

WAC Word Online Web Services Collaboration Team

Microsoft *Redmond, WA*

Software Engineer Intern **June 2017-August 2017**

Yammer Back-End Web Services Growth Initiative

Microsoft *San Francisco, CA*

Software Engineer Intern **June 2016-August 2016**

Kindle Wholesale Devices Experience Developer

Amazon *Seattle, WA*

RESEARCH EXPERIENCE

Graduate Research Assistant **September 2018–May 2019**

Chapman University *Orange, CA*

Undergraduate Research Assistant **September 2015–May 2018**

Chapman University *Orange, CA*

TEACHING EXPERIENCE

Graduate Teaching Assistant **September 2018–May 2019**

Chapman University *Orange, CA*

LIST OF PUBLICATIONS

Abigail Sandra Atchison

Ott, J., Atchison, A., Linstead, E. **Exploring the Applicability of Low-Shot Learning in Mining Software Repositories** **May 2019**
Journal of Big Data

Ott, J., Atchison, A., Harnack, P., Bergh, A., Linstead, E. **A Deep Learning Approach to Identifying Source Code in Images and Video** **May 2018**
2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)

Ott, J., Atchison, A., Harnack, P., Best, N., Anderson, H., Firmani, C., Linstead, E. **Learning Lexical Features of Programming Languages From Imagery Using Convolutional Neural Networks** **May 2018**
Proceedings of the 26th Conference on Program Comprehension (ICPC)

Atchison, A., Anderson, H., Berardi, C., Best, N., Firmani, C., German, R., Linstead, E. **Poster: A Topic Analysis of the R Programming Language** **May 2018**
2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)

Stevens, E., Atchison, A., Stevens, L., Hong, E., Granpeesheh, D., Dixon, D., Linstead, E. **A Cluster Analysis of Challenging Behaviors in Autism Spectrum Disorder** **December 2017**

2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)

Atchison, A., Berardi, C., Best, N., Stevens, E., Linstead, E. **A Time Series Analysis of TravisTorrent Builds: To Everything There is a Season** **May 2017**

2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

ABSTRACT

Classifying Challenging Behaviors in Autism Spectrum Disorder with Neural Document Embeddings

by Abigail Sandra Atchison

The understanding and treatment of challenging behaviors in individuals with Autism Spectrum Disorder is paramount to enabling the success of behavioral therapy; an essential step in this process being the labeling of challenging behaviors demonstrated in therapy sessions. These manifestations differ across individuals and within individuals over time and thus, the appropriate classification of a challenging behavior when considering purely qualitative factors can be unclear. In this thesis we seek to add quantitative depth to this otherwise qualitative task of challenging behavior classification. We do so through the application of natural language processing techniques to behavioral descriptions extracted from the CARD Skills dataset. Specifically, we construct 3 sets of 50-dimensional document embeddings to represent the 1,917 recorded instances of challenging behaviors demonstrated in Applied Behavior Analysis therapy. These embeddings are learned through three processes: a TF-IDF weighted sum of Word2Vec embeddings, Doc2Vec embeddings which use hierarchical softmax as an output layer, and Doc2Vec which optimizes the original Doc2Vec architecture through Negative Sampling. Once created, these embeddings are initially used as input to a Support Vector Machine classifier to demonstrate the success of binary classification within this problem set. This preliminary exploration achieves promising classification accuracies ranging from 78.2-100% and establishes the separability of challenging behaviors given their neural embeddings. We next construct a multi-class classification model via a Gaussian Process Classifier fitted with Laplace approximation. This classification model, trained on an 80/20 stratified split of the seven most frequently occurring behaviors in the dataset, produces an accuracy of 82.7%. Through this exploration we demonstrate that the semantic queues derived from the language of challenging behavior descriptions, modeled using natural language processing techniques, can be successfully leveraged in classification architectures. This study represents the first of its kind, providing a proof of concept for the application of machine learning to the observations of challenging behaviors demonstrated in ASD with the ultimate goal of improving the efficacy of the behavioral treatments which intrinsically rely on the accurate identification of these behaviors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Challenging behaviors are defined as behaviors that are not culturally or socially acceptable, and can put the physical safety of the individual and/or others in jeopardy, affect learning, and limit access to community settings [1, 2, 3]. These behaviors often affect an individual's ability to interact with their environment consistently and positively [4, 5, 6]. Individuals with Autism Spectrum Disorder (ASD) have been found to demonstrate such challenging behaviors with significant frequency [4]. Further, the severity and prevalence of challenging behaviors in the lives of individuals with ASD has not been found to be limited to one specific developmental stage such as adolescence or early childhood [7, 2, 8]. In short, improving the efficacy and efficiency of the treatment of ASD means that researchers must look to better understand challenging behaviors exhibited by individuals with ASD in order to best leverage and target behavioral treatments.

In this thesis we look to explore the application of natural language processing and supervised machine learning to the problem of classifying challenging behaviors exhibited in ASD. By demonstrating that neural word embeddings can be used to differentiate between challenging behaviors, we present the potential for classification models to be deployed in a clinical environment to aid in establishing the context of behavioral treatment.

A common practice in the treatment of behavioral patterns in ASD is Applied Behavior Analysis (ABA), which employs Behavior Analysis in order to develop a specific therapy regimen. Behavior Analysis, defined as the scientific study of behavior [9], is leveraged by ABA as a means by which an individual's behavioral patterns can be shaped through the consistent application of reinforcement learning and controlling of environmental factors. During ABA, challenging behaviors exhibited are analyzed and defined on a case by case basis. This approach takes into consideration the reality that the same challenging behavior, for example Aggression, can manifest differently across individuals as well as within a single individual over the course of treatment [1, 10]. ABA can be facilitated in a variety of settings, such as home sessions, clinic visits, or even in a school setting. At its core, it seeks to deliver a daily therapy regimen targeting both the behavioral strengths and weaknesses of the individual being treated; a goal that has been demonstrated to provide measurable improvements in the outcome of children on the spectrum when early, intensive (30 or more hours per week) intervention is facilitated [11].

While ABA represents a demonstrably successful way of addressing challenging behaviors, the means by which these behaviors are identified and tracked are highly personalized and these identifications can only be made within the scope and experience of the particular specialist conducting the assessment. We present this thesis with the hopes of adding quantitative depth to this process through behavioral classification.

The potential to leverage neural word embeddings constructed from behavioral descriptions to classify challenging behaviors holds great value not only in the understanding of these behaviors but also in clinical applications. Through an exploration of the models constructed, we demonstrate this potential by outlining the means by which quantitative techniques can be applied to highly qualitative data in order to better guide the identification of behaviors exhibited over the course of ABA treatment. In the pages that follow, we will detail the data used in our study (chapter 2), the machine learning methods leveraged in the construction and modeling of word embeddings (chapter 3), results obtained (chapter 4), and their practical significance (chapter 5). We then outline future work and previous applications (chapter 6) along with final conclusions (chapter 7).

Chapter 2

Data

To determine if neural word embeddings could aid in the classification of challenging behaviors, we started with an analysis of behavioral descriptions across 15 behaviors. The descriptions were provided by the Center for Autism and Related Disorders (CARD), one of the largest national providers of ABA therapy services.

The dataset, known as the CARD Skills^{TM} dataset, is a clinical database that houses the ABA curriculum for patients, as well as a detailed log documenting the evolution of patient progress from the start of services to termination of services, which typically spans several years. In this study we focus specifically on recorded challenging behaviors and the description associated with that particular demonstration of the behavior recorded. In the case of the CARD skills dataset, all patients were under direct supervision of BIs and BCBAs employed by CARD during treatment.

The dataset consists of treatment history for 1,602 individuals. After an initial preprocessing step to remove all challenging behavior labels which did not have descriptions associated with them, the resultant dataset consisted of 1,917 total observations of challenging behaviors during ABA treatments. These descriptions were then tokenized using the tokenize package provided by the Python Natural Language Toolkit [12]. Following tokenization, each row in the dataset corresponds to a single instance of a challenging behavior. That instance consists of the label of the challenging behavior being exhibited and an ordered collection of tokens associated with that label, with each token being a word from the original description.

Behavior	**Count**
Aggression	462
Disruption	140
Elopement	64
Hoarding	3
Inappropriate Sexual Behavior	11
Lying	5
Noncompliance	221
Obsessive Behaviors	27
Pica	10
Self-Injurious Behavior	88
Stealing	1
Stereotypy	188
Tantrums	252
Teasing/Bullying	11
Other	438

Table 2.1: Challenging Behavior Frequency Counts

These behavioral labels are one of 15 potential categorizations. The frequency counts of each of these categorizations in the dataset can be found in Table 2.1. One count corresponds to one row in the dataset indicating a description of an action reported during an ABA therapy session that was subsequently labeled as that challenging behavior. Due to the variance of sample size across behaviors, in our model construction we solely focused on the classification of the 7 most commonly occurring behaviors in the dataset. These behaviors were Aggression, Disruption, Elopement, Noncompliance, Self-Injurious Behaviors, Sterotypy, and Tantrums. A definition for each of these select behaviors derived from previous literature can be found in Table 2.2.

In addition to the behaviors studied in this analysis, it should be noted that behavioral instances listed as "Other" are numerous in the dataset. While we do not seek to analyze these descriptions in this particular study, their prevalence in the dataset further contributes to the potential value of a classification model as a means to reduce the occurrence of these ambiguous labels.

Behavior	**Definition**
Aggression	Generalized hitting, kicking, biting, punching, scratching and throwing (i.e. furniture)[13]
Disruption	Hitting, kicking, biting, punching, scratching and throwing (i.e. furniture) in response to something non-specific to the individual[13]
Elopement	Wandering, leaving, and running from safe spaces or adult supervision[14]
Noncompliance	Any behavior other than what has been requested within a specified period of time[15]
Self-Injurious Behavior	Behaviors in which individuals cause physical damage to his or her own body (scratching, biting, head banging, chin hitting, hair pulling, skin picking, eye pressing or gouging[16, 17]
Stereotypy	Repetitive and non-functional behavior[18, 19]
Tantrums	Hitting, kicking, biting, punching, scratching and throwing (i.e. furniture) in response to specific to the individual[13]

Table 2.2: Challenging Behavior Definitions

Chapter 3

Methods

To analyze behavioral descriptions in our data we employ two different natural language processing techniques in order to construct neural word embeddings. The vectorization of a natural language data collection adds quantitative depth to otherwise flat, textual data, allowing for the application of machine learning algorithms.

The first technique used in this thesis is Word2Vec. At its core Word2Vec is the construction of a vector space that models the semantic and syntactic meaning of words found within a textual corpus. These vectors are generated through the training of a neural network either via the Skip-Gram approach or the Continuous Bag of Words (CBOW) approach. In either approach, word vectors are generated through back-propagation over a weight matrix after predictions are made given a word and the context in which that word appears in the text corpus. In this way, Word2Vec derives the deeper meaning of a word based off of the idea best summarized by the English linguist J.R. Firth: ”You shall know a word by the company it keeps”.

Following the construction of this vector space, the term frequency - inverse document frequency (TF-IDF) score for each word in a given document is combined with its corresponding word vector representation in order to construct weighted document vectors. TF-IDF is a commonly used technique in NLP as it considers the overall frequency of a word in the entire corpus, or its term frequency, as well as the frequency of that word in the specific document we are currently generating a vector for, its document frequency. If a word appears more frequently in the corpus this will lower its TF-IDF score as that word is perhaps a more general stop word. Inversely, if that word appears frequently in the current document, this will increase its TF-IDF score as this frequency could indicate the importance of that particular word to the current document. These scores are then used to generate weighted document vectors for each behavior report by taking the TF-IDF weighted sum of the neural embedding of each word that appears in a given document.

The second technique used to analyze behavioral descriptions, Doc2Vec, expands on the Word2Vec paradigm while eliminating the need for the final step of generating document vectors through a weighted sum. Doc2Vec again looks to add quantitative depth to textual data through the repeated adjustment of a weight matrix. However rather than training exclusively on the words in our corpus and the context in which they appear, it also learns the neural embeddings of individual documents by using them as a feature in the weight matrix used to make decisions in the model. In the scope of this study the documents being modeled are the single behavioral descriptions recorded during ABA treatment.

Once these neural embeddings are learned, they can be used as input data in supervised machine learning algorithms in order to demonstrate whether or not they contain enough information to differentiate between behaviors. Commonly referred to as classification models, supervised learning learns the means by which a label can be produced given an input through exposure to already labeled data. At their core, most classification algorithms operate by minimizing incorrect label assignment through an adjustment of features used to predict a label given an input. These features can be as complex as a deep weight matrix or as simple as coefficients in a single, linear function. While the specifics to the construction of classification models varies, their overall mission remains consistent: given truth data consisting of inputs and expected labels for those inputs, learn features that will allow us to correctly classify data we have not seen before.

For this analysis we consider two algorithms to extract meaningful features from our data. The first, the Support Vector Machine (SVM), is a algorithm which looks to fit a hyperplane to a set of input data by maximizing the margin that separates classes within the dataset in order to differentiate between possible labels [20]. In this study we used SVMs as a means to produce binary classifications. This baseline algorithmic analysis of the document vectors generated previously serves to demonstrate the potential of studying challenging behaviors using neural embeddings. We then move to a Gaussian Process Classifier in order to build a classification model which differentiates beyond binary classification, in this case a 7-class output. Our Gaussian Process uses Laplace approximation to smooth several one-class-versus-rest Gaussian Distributions which can then be sampled to determine the final classification of any one input [21].

In the following subsections we provide a brief mathematical overview of Word2Vec, Doc2Vec, SVM, and Gaussian Process Classification using Laplace approximation. We then apply these algorithms and analyze the results in the remaining chapters.

3.1 Word2Vec

Consider a corpus of word vectors V, in which each word, i, in the corpus is represented by a unique one-hot vector v_i. Word2Vec takes this collection of words and generates a new matrix, W, in which each word, i, in the corpus is now represented by a rich word embedding w_i which represents the semantic meaning of that word as it is used in the given corpus.

This matrix, W, is generated through propagation over a weight matrix in an architecture similar to that of a feed-forward Neural Net Language Model where the non-linear hidden layer has been removed and the projection layer, W, is shared for all words [22].

The training of the parameters in this modified neural net is handled via two different approaches in Word2Vec. In both architectures, described below, the resultant word embeddings, W, are trained using stochastic gradient descent via back-propagation. After convergence is achieved these embeddings represent the semantic context in which each word in our corpus appears. Thus, words used in similar contexts will be mapped to similar locations in the vector space constructed (i.e. ”run” and ”walk” might appear near each other due to semantic similarity while ”run” and ”fast” would appear in different areas in the vector space because, despite common co-occurrence, they are semantically used differently). The word embeddings thus give rich, quantitative depth to otherwise flat, textual data and can be used as input to machine learning algorithms with great promise.

3.1.1 Continuous Bag of Words (CBOW)

The first architecture used to train the weight matrix, W, which will eventually become our neural word embeddings is the Continuous Bag of Words (CBOW) approach. In this methodology, a window of size C is shifted over the corpus. For each word, v_i, in the current context window, the activated index in the one-hot embedding for that word will map to a specific column in the weight matrix, W. This vector, w_i, contributes to a summed projection which is used to calculate the word which appears in the middle of the current context. Thus, for a window of size C we let $k = C/2$ and use the k words before and after our target word to determine this target.

In short, the goal of training this model, given a sequence of untrained word embeddings W such that $W = \{w_1, w_2, ..., w_T\}$, is to maximize the average log probability of predicting a word given the context, C or $k \times 2$, in which it appears

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p(w_t|w_{t-k}, ..., w_{t+k})$$

The prediction is done via a multi-class softmax classifier in which

$$p(w_t|w_{t-k}, ..., w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

Each y_i is an un-normalized log-probability for each output word, i, computed as

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W)$$

3.1.2 Skip-Gram

In the second architecture a weight matrix, W, representing the word embeddings of our corpus, V is still produced. However, while CBOW takes the context $k \times 2$ or $w_{t-k}, ..., w_{t+k}$ in which a word appears as input and predicts that word, w_t, Skip-Gram attempts to maximize the classification of a word, w_t, based on another word in the context, $k \times 2$ or $w_{t-k}, ..., w_{t+k}$. In short, for Skip-Gram the target word, w_t is used as input and the context of that word, $w_{t-k}, ..., w_{t+k}$, is predicted. This is achieved by using each word in the corpus as an input to a log-linear classifier with a continuous projection layer as output, in order to predict words that appear within the defined context, $C = k \times 2$.

While increasing the size of the window does positively affect the quality of resulting word embeddings, it also contributes to computational complexity. It should also be noted that at training time, less weight is given to correctly predicting words that appear farther away from the input word by sampling less of those words in training examples.

3.1.3 TF-IDF Scoring

After word embeddings are generated, a TF-IDF score $TF - IDF_{i,j}$ for each word i in each document, j, in the corpus can be defined as:

$$TF - IDF_{i,j} = (tf)_{i,j} \times \log(T/df_i)$$

This weight is then used to create weighted document vectors made up of the vector sum of all word embeddings in a specific document multiplied by their corresponding TF-IDF score.

3.2 Doc2Vec

The Paragraph Vector algorithm, introduced by Le et al. and implemented as Doc2Vec, is an unsupervised algorithm which learns a fixed-length feature representation of documents through representing each document as a dense vector which is then trained to predict words in the document [23]. For the remainder of this thesis these produced paragraph vectors will be referred to as document vectors.

This approach derives its origins in word vector learning, namely the CBOW method, described above, in which word vectors are used as a way to predict the next word in the sentence, effectively capturing the semantic meaning of words in the produced embeddings during training despite starting with randomly initialized vectors. This same methodology is mirrored in the Paragraph Vector algorithm, except document vectors are now included as a feature in the prediction of the next word given a context sampled throughout the document. This addition allows for document vectors to be trained in the same way word vectors are learned.

Thus the objective of the Doc2Vec model is still to maximize the average log probability equation defined as

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k},...,w_{t+k},d_a)$$

in which we are predicting the word,w_t, which appears in document,d_a given the context in which that word appears, $w_{t-k},...,w_{t+k}$. However, now a new feature has been introduced to aide in this prediction: the document vector, d_a which corresponds to the document from which this word and context have been sampled.

The prediction task is then conducted via a multi-class softmax classifier in which

$$p(w_t|w_{t-k}, ..., w_{t+k}, d_a) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

In this, each y_i is an un-normalized log-probability for each potential output word computed as

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W, d_a; D)$$

Where U and b are softmax parameters and h is constructed from a word matrix W which represents a collection of vectors, $W = \{w_1, w_2, ..., w_q\}$ in which every vector maps to a unique word in the corpus and a matrix D which represents a collection of vectors, $D = \{d_1, d_2, ..., d_p\}$ in which every unique vector maps to a document in the corpus.

By this design, a document embedding learned by this model can be thought of as another word in the Word2Vec CBOW model, and acts as memory that tracks what is still yet to be modeled in the current context of the document.

These contexts used at training time are of fixed length and extracted via a sliding window of size $k \times 2$ applied to each document. The document vector produced, d_a, representing a single column in the matrix D, is shared across context for the local document d_a currently being sampled. However, these contexts are not shared across documents in the entire corpus. Dissimilarly, the word vector matrix, W, is shared across documents meaning that the semantic meanings of words are learned and applied at the generalized, corpus level rather than at the document-specific level, adding generalizability to the model.

The vectors in W and D are learned during training using stochastic gradient decent obtained via back-propagation across the two weight matrices. Thus, if there are N documents in the corpus, M total words in the vocabulary, the model is aiming to learn document vectors of dimension q, and each word is mapped to the dimension p, the model will be trained across a total of $N \times q + M \times p$ parameters.

A core strength of this algorithm is that the input documents can be of variable length whereas in previous models, namely word2vec, the window used in training is kept static rather than considering the varying length of input word vectors.

After training, these document vectors can then be used as input to a variety of machine learning algorithms. Such algorithms will be explored in the remainder of this chapter and the application of these document vectors to these algorithms will be discussed in Chapter 4.

3.2.1 Negative Sampling

To address the large number of parameters to be adjusted during training time, negative sampling has been proposed as an optimization over the softmax layer generally used to determine outputs [22]. In the original model the objective function seeks to maximize the log probability of predicting a word or context (w_O), given an input (w_I) word or context, depending on the architecture being implemented. With negative sampling, the objective function instead looks to maximize the dot product of w_O and w_I while also minimizing the dot product of w_I and randomly sampled "negative" words.

Negative sampling has been found to improve efficiency at training time while also producing better word vectors on average [22]. Nonetheless, our dataset consists of documents made up of fewer words than those used as benchmarks in previous evaluations. As such, eliminating training adjustments to all but the negative and positive vectors results in too drastic of a loss in information for this particular dataset due to the size of each document being learned. Due to this loss, Doc2Vec embeddings constructed using the default softmax output at training time outperform those created using Negative sampling in our classification models, as discussed in Chapter 4.

3.3 Support Vector Machines

Upon construction of vector embeddings, consider a data matrix, D, of dimension $m \times n$. D can then be represented as a collection of vectors, $D = \{X_1, X_2, ..., X_m\}$. Each vector, X_i, represents a unique data instance, in this case a document embedding, and each vector element, $X_{i,j}$, a specific measurement (attribute) for that embedding.

An SVM classifier takes in D as input and outputs a set of weights $W = \{w_1, w_2, ..., w_n\}$, one for each feature, m, in the input such that for each vector X_i its corresponding label y_i will be 1 for positive samples and -1 for negative samples. Thus, we can state that

$$y_i(W \cdot X_i + b) - 1 \geq 0$$

The linear combination of X_i and W predicts the class label, y_i for any given data point X_i. These weights are determined through the approximation of the best hyperplane which maximizes the margin between two classifications. This margin, L, can be defined as:

$$L = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j X_i \times X_j$$

Upon maximizing L for the training inputs we are left with our weight vector, W. Given these weights, W, and an unknown data points, u, if $W \cdot u + b \geq 0$ then u is a positive classification.

While SVMs provide an algorithmically simple way to differentiate between two classes in a convex space which avoids local maxima, when data points are not linearly separable the mechanism struggles. This can be addressed by introducing a Kernel method K which represents the dot product between two input vectors into a new space such that $K_{X_i,X_j} = \phi(X_i) \cdot \phi(X_j)$. Some common Kernels include linear and radial basis kernels. In this thesis each SVM model is fitted using a radial-basis function (rbf) as the kernel.

3.4 Gaussian Process Classifier

Again, consider a data matrix, D, of dimension $m \times n$. D can then be represented as a collection of vectors, $D = \{X_1, X_2, ..., X_m\}$. Each vector, X_i, represents a unique data instance, and each vector element, $X_{i,j}$, a specific measurement (attribute) for that point.

A Gaussian Process defines a non-parametric distribution over functions, $p(f)$, such that for any finite subset $f_{s-w}\{f_s, f_{s+1}, ..., f_{w-1}, f_w\} \subset D$ the marginal distribution over that subset has a multivariate Gaussian distribution.

In the Gaussian Process Classification (GPC) used in this thesis, during training a one versus rest Gaussian Distribution for each of the C classes, $C_1, C_2, ..., C_C$ is constructed. These distributions model the trend of the data, D, while maximizing the space between classes, C, and, once trained, can be leveraged in classification. Given a previously unseen input vector X_i we determine the most likely classification by sampling our constructed Gaussians and choosing the closest fit. While computationally expensive to train due to GPC's foundation in Bayesian inference, when applied to small, unbalanced datasets, like the one studied in this thesis, this method models multiple classes with great success. For an in-depth treatment of GPC see work outlined by Rasmussen et al. [21].

3.4.1 Laplace approximation in GPC

It should be noted that in this function approximation the approximate log marginal likelihood across multiple classes combined with the Gaussian prior is non-Gaussian and thus the resultant non-Gaussian posterior process must be smoothed using Laplace approximation [21].

Chapter 4

Results

Previous analysis of the topography of challenging behaviors have manifested as largely qualitative in nature [3]. These studies effectively underline the difficulty as well as the necessity of the consistent and reliable labeling of challenging behaviors exhibited by individuals with ASD. In this labeling, one must distinguish between behaviors which, at times, can only be separated based on the implied function of these actions as is the case with Disruptions and Tantrums [3]. In this thesis we look to diverge from a qualitative analysis into the quantitative, allowing for the discovery of patterns and features in behavioral data that could only be extracted through machine learning algorithms. Below we will discuss the results of applying the methods examined in Chapter 3 to the challenging behavior descriptions and their labels introduced in Chapter 2. We present these findings in this chapter with the hope that in the future clinicians can use feedback from similar models to identify the most likely behavioral label given actions exhibited in therapy sessions. For a more in depth treatment of the statistical underpinnings of these methods, refer back to Chapter 3.

4.1 Construction of Neural Document Embeddings

In analyzing the CARDS skills data of challenging behavior descriptions and their labels, we constructed a neural embedding for each challenging behavior report using three methods: Doc2Vec with negative sampling, Doc2Vec with hierarchical softmax, and a TF-IDF weighted sum of Word2Vec word vectors. Each set of embeddings serve as input to the classification algorithms discussed in Chapter 3, namely Support Vector Machines and Gaussian Process Classifiers. All computations were done using the computing software Python.

The creation of Word2Vec embeddings on behavioral descriptions was conducted using the gensim "word2vec" package [24]. The embeddings were produced using the Skip-Gram architecture. The optimal embedding size to allow for sufficient information capture was empirically chosen to be 50. In order to ensure that instance specific words do not affect behavioral classification a minimum frequency of 5 was set, meaning that a vector embedding was only constructed for a word if it appeared at least 5 times across the corpus. From scikitlearn, the "TfidfVectorizer" was used to generate TF-IDF scores for each of the tokenized words in the corpus [25]. In order to generate the weighted Word2Vec embeddings, for each row in the dataset the Word2Vec modeled was queried for that word and multiplied by it's corresponding TF-IDF weight. This product was then summed across all of the words in that specific row and the final sum was the resultant document vector.

The creation of Doc2Vec document embeddings was done using the gensim "doc2vec" package [26]. These embeddings were made to be vectors of size 50 in order to match those created in our weighted Word2Vec process and were constructed with a minimum word frequency count of zero, as in this process we are not directly summing the word vectors in the construction of the document vectors. For the Doc2Vec model trained with negative sampling our negative word count was 5. For both Doc2Vec models the initial learning rate α was set to 0.025 with αmin set to 0.001, the optimal parameter settings empirically determined for this architecture by [27].

4.2 Analysis of Neural Document Embeddings

Once the construction of 3 document embeddings was completed, the information captured by these embeddings was evaluated in order to determine if this method can be used to successfully differentiate between challenging behaviors in ASD. First, binary classification of the document embeddings was conducted using SVMs. A multi-class classification model was then constructed using GPC. In both of these explorations, the test accuracy resultant from classification models built on an 80/20 training/testing split was used as a metric for effectiveness of neural embedding. Ultimately, while all three embedding types were found to hold a depth of information, the Doc2Vec embedding trained using hierarchical softmax was determined to be the most effective in classification. These classifications will be discussed in the sections to follow.

4.2.1 Binary Classification using Support Vector Machines

The first classification model applied to each of these embeddings was a Support Vector Machine (SVM) using a Radial Basis Function kernel. To apply the SVM classifier to the document vectors for the purposes of binary classification, we use the "SVC" class in the "sklearn" Python package [28]. Table 4.1 outlines the results of each unweighted, binary SVM classification. For each classification, the model was trained on an 80/20 stratified split of the weighted W2V, D2V with Negative

Sampling, and D2V with Hierarchical Softmax document embeddings respectively. The highest classification accuracy achieved on these three separate inputs has been bolded; in cases where two embedding input types produced equal test accuracy both metrics have been bolded.

Class A	Class B	Test Accuracy
Elopement	Self-Injurious Behavior	96.77, **100**, **100**
Aggression	Noncompliance	97.8, 94.1, **99.3**
Aggression	Elopement	97.1, 92.4, **97.1**
Noncompliance	Self-Injurious Behavior	95.16, 93.5, **96.8**
Sterotypy	Self-Injurious Behavior	87.5, 92.9, **96.4**
Elopement	Stereotypy	88.2, 88.2, **96.1**
Aggression	Disruption	92.5, 93.3, **95.8**
Elopement	Tantrums	93.3, 90.5, **95.6**
Aggression	Stereotypy	93.1, 93.1, **94.6**
Sterotypy	Tantrums	93.2, **94.3**, **94.3**
Noncompliance	Stereotypy	89.0, 92.7, **93.9**
Disruption	Self-Injurious Behavior	**93.5**, 89.1, **93.5**
Disruption	Noncompliance	87.5, 84.7, **93.1**
Aggression	Tantrums	88.0, 90.8, **93.0**
Noncompliance	Elopement	88.7, 84.2, **91.2**
Aggression	Self-Injurious Behavior	83.6, 86.3, **90.9**
Disruption	Elopement	**90.2**, 80.5, **90.2**
Disruption	Stereotypy	81.8, 87.9, **89.4**
Noncompliance	Tantrums	81.9, 84.0, **87.2**
Disruption	Tantrums	66.7, 70.5, **78.2**

Table 4.1: SVM Classifier Results sorted by test accuracy
Test Accuracy displayed is for inputs of TF-IDF Weighted Vec, Doc2Vec with Negative Sampling, Doc2Vec with Hierarchical Softmax respectively. The highest classification accuracy has been bolded.

These accuracy metrics make clear that there is separation between neural embeddings given any two behavioral classes in our dataset. This bears particular promise as previous qualitative analysis of these descriptions has highlighted the difficulty of separating challenging behaviors when considering the topography of these descriptions.

Hong et al. used a part-of-speech analysis on bag-of-words representations of challenging behavior descriptions to identify the top 20 verb terms used to define the topography of each challenging behavior[3]. Their findings, in particular the qualitative overlap between challenging behaviors when studying the verbiage used in clinical descriptions, can be used as a counter-point to our quantitative analysis of the vector space created by neural embeddings constructed from similar descriptions.

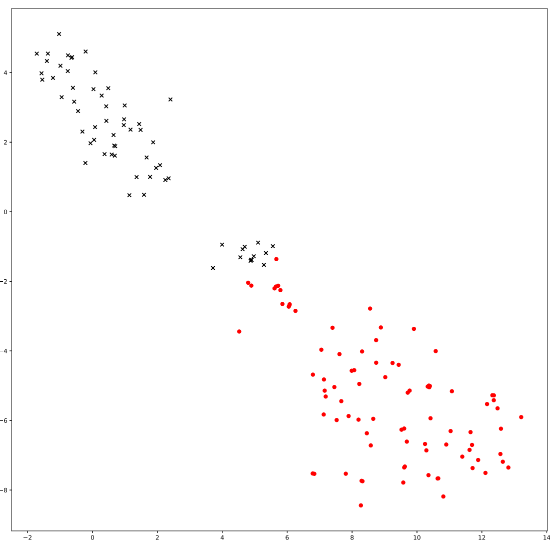
Self-Injurious Behavior (SIB) and Aggression shared 38.5% overlap in their descriptional verbiage. This is unsurprising when one considers these behaviors only truly diverge in regard to their target, SIB targeting onesself while Aggression targets an outside entity. Our model distinguishes between these behaviors with a testing accuracy of 90.9%.

Noncompliance and Tantrums also bore many similarities in the verbiage used to describe them in a clinical setting. Specifically, 37.9% of the 20 most common verb terms used in describing them were shared between these two behaviors. Our SVM model distinguishes between the two behaviors with a test accuracy of 87.2%.

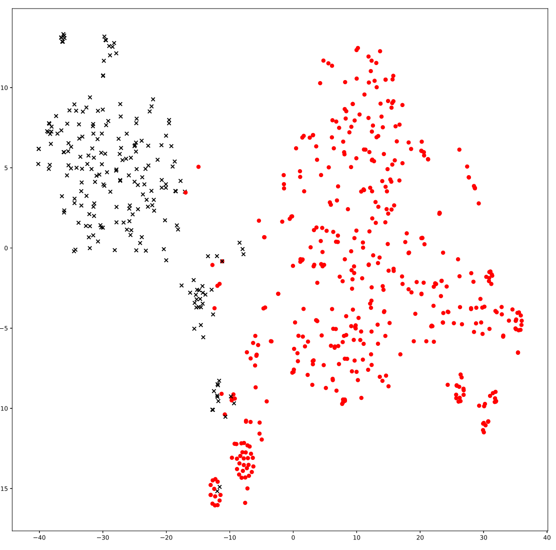
Additionally, the difficulty of qualitatively distinguishing between Aggression, Disruption, and Tantrums was presented in that they overlapped by 33%, sharing ll of 20 most common verb terms used to describe them including crying, screaming, throwing, and protesting. In our results the classification accuracy achieved when fitting a hyperplane to an Aggression vs. Tantrum and Aggression vs. Disruption vector space was 93.0% and 95.8% respectively. The worst accuracy presented by the pairwise binary classification between these three classes was 78.2%, which was achieved when classifying between disruption and tantrums. This classification accuracy provides further insight to the shared action between Disruptions and Tantrums. The process of demonstrating disruptive behavior is often the same process followed in the manifestation of a tantrum, an empirical truth reinforced by the interchangeable use of these terms in the behavioral analysis of ASD,[3], as well as in the finding that of the top 10 verbs used to describe these two behaviors in a clinical setting, four were shared between disruption and tantrums (crying, screaming, throwing, and yelling). Thus, our hypothesis holds true in that a quantitative analysis of challenging behaviors, even those found to be qualitatively indistinguishable at times, will produce deeper insight and perhaps provide a mechanism to aide clinicians in their analysis of behavioral manifestations during treatment.

In order to further demonstrate the linear separability of the neural embeddings constructed in this analysis a pairwise comparison of skills has been visualized for the four most successful classifications (see Figure 4.1) and the four least successful classifications (see Figure 4.2) in our SVM model. Here, given the document embeddings of two challenging behaviors, t-Distributed Stochastic Neighbor Embedding (t-SNE) was used to reduce the 50-D vector space constructed to a 2-D, graphically-feasible entity. The document vector space used in this reduction and visualization was generated by doc2vec trained using hierarchical softmax as an output layer.

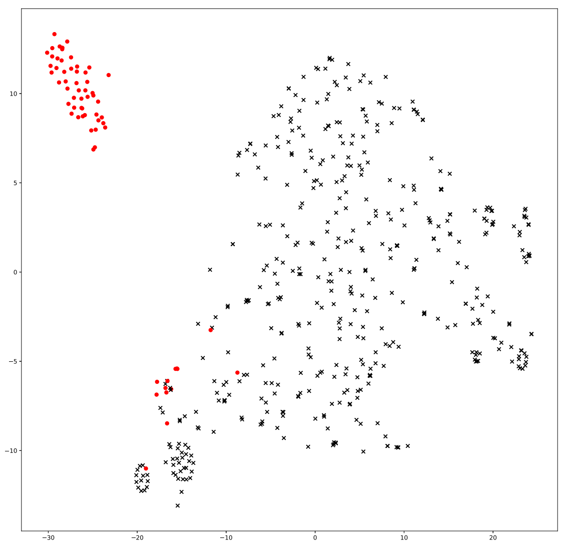
Per these visualizations we can see that challenging behaviors are distinguishable in our constructed vector space, even when the frequency of classes is unbalanced in our dataset. This separation is universal across each binary classification presented, whether the classification accuracy was amongst the best or the worst scoring, with the exception of Disruption vs Tantrums seen in Figure 4.2(a). In this plot, separation is inconsistent and the resultant classification accuracy of 78.2% can be expected upon inspection.



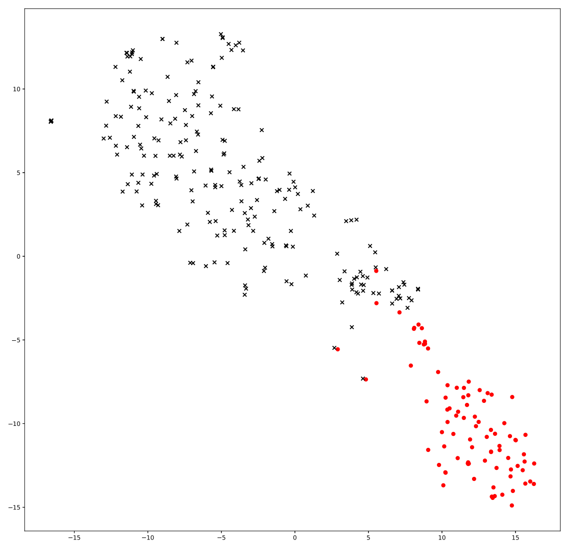
(a) Elopement vs SIB - 100%



(b) Aggression vs Noncompliance - 99.3%

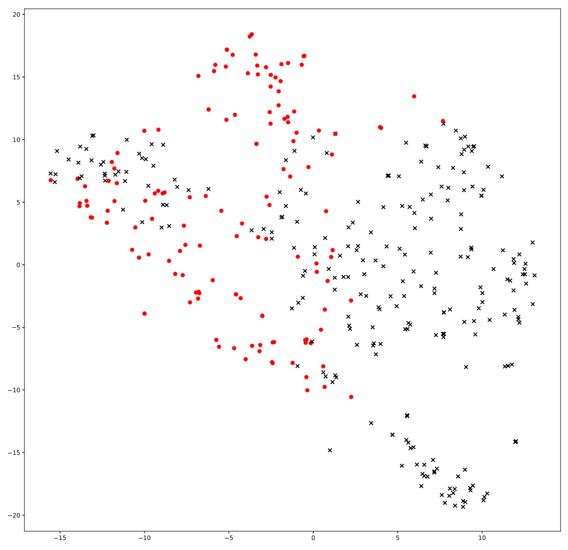


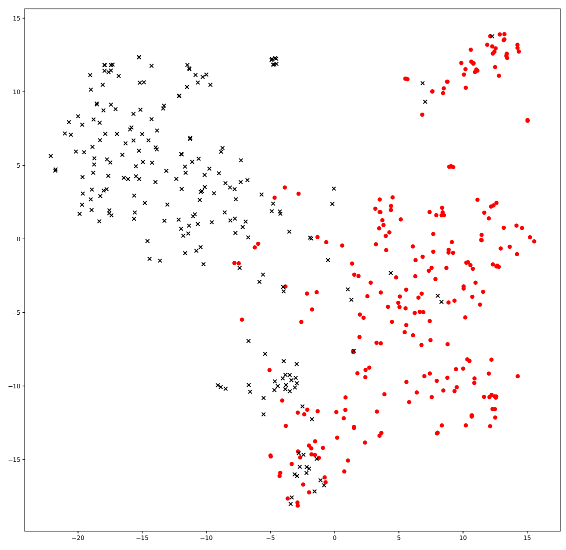
(c) Aggression vs Elopement - 97.1%

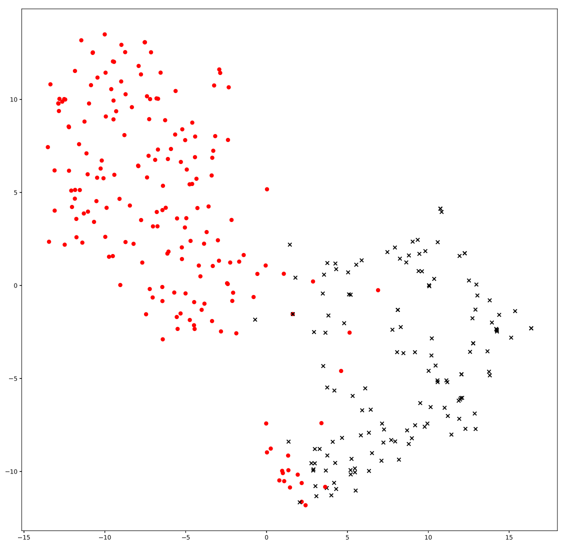


(d) Noncompliance vs SIB - 96.8%

Figure 4.1: Visualization of the four most accurate binary classifications and their corresponding test accuracy.
The SVM model used in classificaiton was trained using our Doc2Vec embeddings built with hierarchical softmax


(a) Disruption vs Tantrums - 78.2%


(b) Noncompliance vs Tantrums - 87.2%


(c) Disruption vs Stereotypy - 89.4%

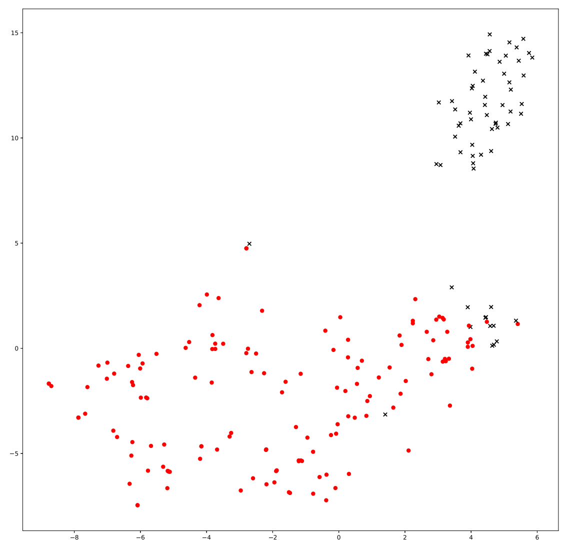

(d) Disruption vs Elopement - 90.2%

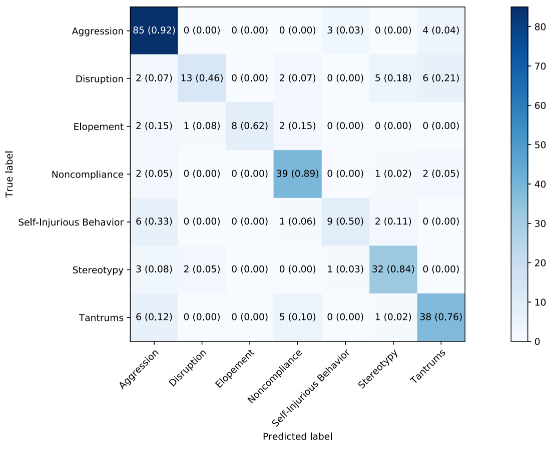
Figure 4.2: Visualization of the four least accurate binary classifications and their corresponding test accuracy.
The SVM model used in classification was trained using our Doc2Vec embeddings built with hierarchical softmax

4.2.2 Multi-Class Classification using Gaussian Processes

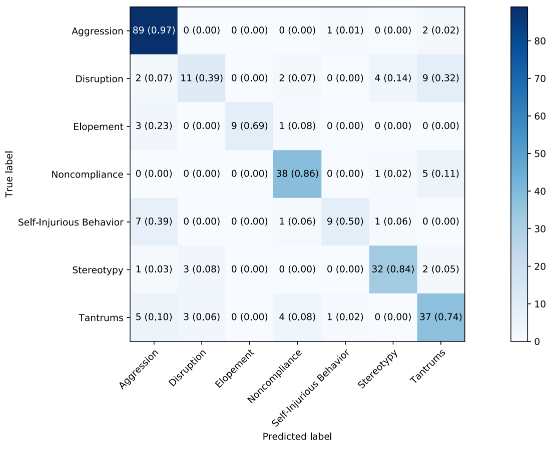
Binary classification within the behavioral dataset provides a promising first analysis of challenging behaviors as well as a counter-point to previous qualitative analysis. Nonetheless, in order to be leveraged in practice a multi-class classification model is needed to differentiate between the many behaviors the could possibly manifest during behavioral treatment.

To achieve this, a Gaussian Process Classifier (GPC) was built to generate classifications across 7 challenging behaviors. This model was constructed using the "GaussianProcessClassifier" function in the scikit learn Python package [29] and was trained on an 80/20 stratified split of each document embedding method. Each model took approximately two minutes to converge. Figure 4.3 displays the confusion matrices produced when using Weighted Word2Vec, Doc2Vec with Negative Sampling, and Doc2Vec with Hierarchical Softmax embeddings as input. The GPC trained on Doc2Vec with Hierarchical Softmax embeddings as input performs the best across all seven classes with a training accuracy of 88.8% and a testing accuracy of 82.7%.

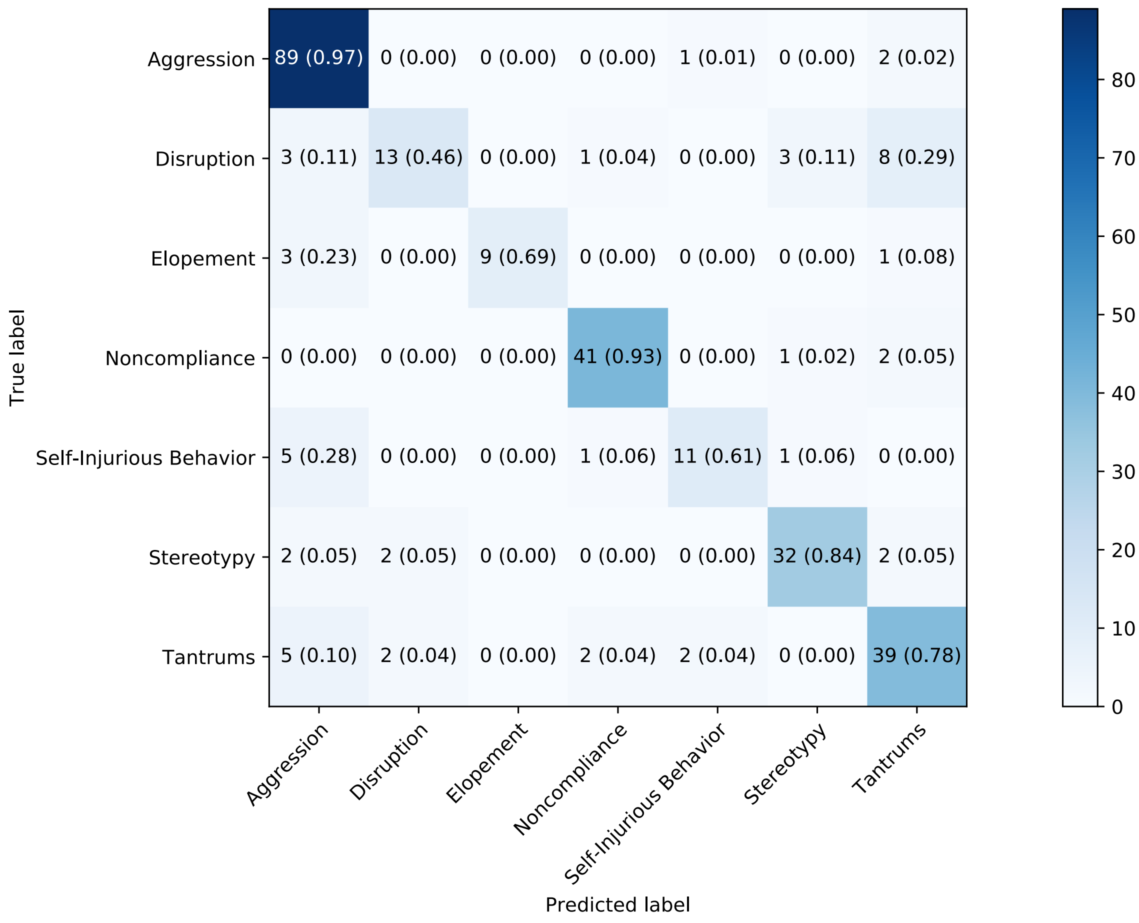
A weakness of the classifier to identify Disruption, Elopement, and Self-Injurious Behavior is revealed in the confusion matrix in Figure 4.3c. These exemplars were classified with 46%, 69%, and 61% accuracy, respectively. In the case of Disruption, this class was misclassified as Tantrums 8 times and Aggression 3 times, a reasonable failing when considering these behaviors overlap in their topographic definition and share common descriptive verbiage [13, 3]. The misclassification of Disruption could be attributed to the ambiguous nature of challenging behavior labeling and the overlap that exists across these behaviors. Elopement and SIB were most commonly misclassified as Aggression, 3 times and 5 times respectively. Again, both of these true behavioral classifications could be seen to manifest as Aggression as well when



(a) TF-IDF Weighted Word2Vec



(b) Doc2Vec with Negative Sampling



(c) Doc2Vec with Hierarchical Softmax

Figure 4.3: The confusion matrices of the classifications produced by a Gaussian Process Classifier trained with three different neural embeddings as input

one considers their generalized definitions. These misclassifications could additionally be contributed to by the frequency of Aggression document vectors in the dataset compared to any other class. In this particular comparison, outweighing SIB 4 to 1 and Elopement by nearly 5 to 1.

Through the analysis of these three neural embeddings via two classification model architectures, we demonstrate that the semantic and syntactic queues extracted during neural embedding construction allows for the differentiation between challenging behaviors. Oftentimes, these differentiations have been made between classes that qualitatively have been found to be nearly indistinguishable.

Chapter 5

Discussion

Currently, there is no cure for ASD; however, early intervention and treatment have been shown to facilitate the future success of a child diagnosed with ASD [30]. Behavioral intervention is largely a part of this equation. In the treatment of ASD, behavioral intervention has been found to reduce the manifestation of challenging behaviors in individuals by 80-90% [4]. In some cases these treatments succeed in virtually eliminating all appearances of challenging behaviors [31, 32]. As such, aiding in the reliability of this treatment by providing a mechanism which lends additional insight into behavioral classifications, holds great value.

A topographical analysis of challenging behaviors has highlighted the ambiguity of behavioral descriptions due to shared similarity in verbiage [3]. Nonetheless, in our analysis of the neural embeddings created from this textual data we were able to effectively and consistently differentiate between challenging behaviors. The importance of our work can thus be highlighted in the provision of a quantifiable mechanism for differentiating between behaviors that have qualitative overlap.

It should be noted that in providing quantifiable separation between behaviors we have also added greater depth to the ongoing discourse surrounding the existence of Disruption and Tantrums as two separate behavioral labels. These labels are believed to describe very similar actions which can only be differentiated by their implied function. As such, consistently labeling these behaviors calls for a functional analysis, a utility that is separate from the behavioral analysis being conducted during ABA therapy. To this end we present our findings, specifically the lack of quantifiable separation between the semantic structures used to describe Tantrums and Disruption, as further evidence that these behaviors should be revisited and a single label should be offered to cover this subset of behavioral patterns.

In addition to providing additional insight to therapists labeling demonstrated behaviors in treatment, the prevalence of behaviors labeled as "Other" within the dataset, which represents the second highest occurring label at 438 instances, provides an interesting aside to the classification presented in this thesis. This demonstrated presence of ambiguity in challenging behavior representation only further reinforces the need for a supplemental, quantitative metric when analyzing behaviors. Additionally it provides an opportunity to give greater depth to existing files by providing a sub-label to behavioral reports already labeled as "Other".

In short, the classification success achieved in this thesis not only demonstrates the quantifiable differences in observational descriptions of challenging behaviors during behavioral therapy but also provides the potential for increasing the efficacy of this crucial form of intervention.

Chapter 6

Related & Future Work

6.1 Creation of Neural Embeddings

The construction of word embeddings in order to produce a richer representation of textual data was first introduced by Bengio et al. in their presentation of a feed-forward neural network language model [33]. Later, Mikolov et al. would introduce the architecture leveraged in this thesis which removes the hidden layer from the NNLM proposed in [33] without significant loss of information in the final embeddings constructed [22]. Other architectures would be proposed such as GloVe embeddings presented in [34].

These word embeddings are often then used in machine learning algorithms, as is done in this thesis, due to the ease with which the vector representations produced can serve as input to many common architectures and applications such as the initialization of neural networks, [35], and machine translation [36]. Additionally, word embeddings have produced significant findings in isolation as they model word to word relationships, a feature leveraged by [37, 38, 39].

Paragraph vectors were first introduced by Mikolov et al. as an improvement to the word2vec paradigm [22]. This architecture learns fixed length vector embeddings of variable length pieces of text or documents. This extension of word embeddings constructs these document embeddings by using them as an additional feature to the weight matrix adjusted at training time and has been widely leveraged since its introduction in a wide-range of domains including automatic video analysis [40], tailored user profile construction [41], and unsupervised classification of sentiment [42].

While doc2vec was the implementation of the architecture introduced in [22] and used in this thesis, others have introduced additional means to extract document embeddings. Kiros et al introduced skip-thought which uses an encoder-decoder neural network architecture to learn dense representations of sentences in a document and then deconstruct this representation to predict words in surrounding sentences [43]. In [44] pp, a mechanism for learning document embeddings based off of paraphase pairs extracted from a large-scale paraphase database, was introduced. While both of these architectures serve a similar purpose as doc2vec, in their review of doc2vec, Lau et al. found that the skip-thought approach under-performed in all applied domains and while pp showed promise in some applications, in studies that require domain specific training of document emebddings, as is the case in this thesis, doc2vec is still most capable of constructing meaningful vector representations of documents in a corpus [27].

6.2 Machine Learning in Autism Spectrum Disorder

To the best of our knowledge this study presents the first application of machine learning to ASD to derive meaning from textual data, however machine learning has been applied to ASD treatment in a broad range of previous studies. In [45] and [46], models capable of aiding in the diagnosis of ASD through a reduction of evaluations necessary to converge on a diagnosis are presented. [47] used SVMs in order to improve the evaluation of individuals progress during the use of ASD intervention tools throughout therapy. [48] and [49] present artificial neural networks able to model the relationship between treatment intensity and outcomes in ASD while [50] applied a random forest classifier to evaluations order to determine case status in Autism Developmental Disability Monitoring.

In ASD diagnosis, the presence of challenging behaviors is not considered a core feature [18]. Nonetheless, an understanding of how challenging behaviors can manifest in individuals with ASD has been the pursuit of many studies. [1] explored contributions to the understanding of the relationship between challenging behaviors and psychiatric disorders through a compilation of current findings related to the assessment and treatment of challenging behaviors. [51] and [4] conducted similarly structured studies in order to quantify the presence and frequency of challenging behaviors in young children with ASD, finding that the prevalence of these behaviors was high across data sources although their findings diverged when quantifying the most commonly tracked behaviors in their respective datasets. In [52] a phenotopic analysis on the challenging behaviors exhibited by individuals participating in ABA therapy was carried out.

While this study was conducted on a similar dataset as the one used in this thesis, Stevens et al. considered behavior frequency counts across patients as their input rather than the particular behavioral descriptions, demonstrating in the phenotypes generated that a mix of challenging behaviors characterize each, but generally a dominant behavioral theme could be extracted. Another in-depth analysis of challenging behaviors [53] aggregated and statistically analyzed challenging behavior data from 22 studies conducted over 20 years predating 2003. In their work, the scarcity of data available to be leveraged in a statistically robust setting was highlighted, further motivating the need for a public ASD dataset in the research community.

When looking to future expansions of the findings outlined in this study, limitations imposed by dataset availability is evident. While the application of deep architectures to classification tasks certainly is not a prerequisite to effective analysis of data[54], the inability to explore these architectures in this analysis due to the low number of digitally recorded behavioral observations is a limitation of our work. In future studies we hope to address this short-coming through the artificial generation of neural document embeddings given the vector space already constructed in this study.

Chapter 7

Conclusion

Despite the significant likelihood that individuals with ASD demonstrate challenging behaviors [4], there is still great variability in the definitions of these behaviors,[55], potentially contributing to degrees of uncertainty when facilitating behavioral interventions. This variability can be attributed to the tendency of challenging behaviors to manifest differently across individuals with ASD and even within individuals over contexts and/or time [1, 10]. This reality is taken into account during ABA therapy in which behavioral definitions are constructed on a case by case basis [9]. Despite this demonstrated variability, in this thesis we show that neural document embeddings are capable of synthesizing clinical descriptions into highly classifiable entities.

Within this thesis, we take the clinical descriptions of challenging behaviors and their corresponding labels and generate neural document embeddings via three methods: TF-IDF weighted word2vec, doc2vec using negative sampling, and doc2vec using hierarchical softmax. The resultant embeddings are then independently analyzed through both binary and multi-class classification via SVM models and multi-class

GPC respectively, the latter of which achieves a classification accuracy of 82.7% over seven behavioral labels.

It should be noted that even as the application of Machine Learning to clinical ASD data continues to gain momentum, the lack of an Internet-scale, public repository of longitudinal data still remains. This dataset could serve as a baseline for more exploratory research in big data. In understanding this it becomes evident that while this thesis lays the foundation for other studies to follow, the most obvious of these future works lies in the continued analysis of challenging behavior descriptions on a larger scale data repository.

In a more general sense, the analysis of textual data curated in the clinical treatment of ASD using neural word embeddings is still a largely unexplored direction of ASD research. The quantitative representations of otherwise highly qualitative information in ASD presents the opportunity to analyze a new dimension of data and, when possible, leverage this analysis to make informed decisions about treatment. The work in this thesis represents a modest contribution toward that goal.

Bibliography

[1] Eric Emerson. *Challenging behaviour: Analysis and intervention in people with learning disabilities.* ERIC, 1995.

[2] Johnny L Matson, Sara Mahan, Julie A Hess, Jill C Fodstad, and Daniene Neal. Progression of challenging behaviors in children and adolescents with autism spectrum disorders as measured by the autism spectrum disorders-problem behaviors for children (asd-pbc). *Research in Autism Spectrum Disorders*, 4(3):400–404, 2010.

[3] Esther Hong, Dennis R Dixon, Elizabeth Stevens, Claire O Burns, and Erik Linstead. Topography and function of challenging behaviors in individuals with autism spectrum disorder. *Advances in Neurodevelopmental Disorders*, 2:206–215, 2018.

[4] Robert H Horner, Edward G Carr, Phillip S Strain, Anne W Todd, and Holly K Reed. Problem behavior interventions for young children with autism: A research synthesis. *Journal of autism and developmental disorders*, 32(5):423–446, 2002.

[5] F Charles Mace, Terry J Page, Martin T Ivancic, and Shirley O'Brien. Analysis of environmental determinants of aggression and disruption in mentally retarded children. *Applied Research in Mental Retardation*, 7(2):203–221, 1986.

[6] Johnny L Matson and Marie Nebel-Schwalm. Assessing challenging behaviors in children with autism spectrum disorders: A review. *Research in Developmental Disabilities*, 28(6):567–579, 2007.

[7] Jane McCarthy, Colin Hemmings, Eugenia Kravariti, Katharina Dworzynski, Geraldine Holt, Nick Bouras, and Elias Tsakanikos. Challenging behavior and co-morbid psychopathology in adults with intellectual disability and autism spectrum disorders. *Research in Developmental Disabilities*, 31(2):362–366, 2010.

[8] Olivia Murphy, Olive Healy, and Geraldine Leader. Risk factors for challenging behaviors among 157 children with autism spectrum disorder in ireland. *Research in Autism Spectrum Disorders*, 3(2):474–482, 2009.

[9] Center for Autism and Related Disorders. ABA resources: What is ABA? `http://www.centerforautism.com/aba-therapy.aspx`. Accessed: 2016-05-26.

[10] Johnny L Matson, David E Kuhn, Dennis R Dixon, Stephen B Mayville, Rinita B Laud, Christopher L Cooper, Carrie J Malone, Noha F Minshawi, Ashvind N Singh, Melissa A Luke, et al. The development and factor structure of the functional assessment for multiple causality (fact). *Research in Developmental Disabilities*, 24(6):485–495, 2003.

[11] O Ivar Lovaas. Behavioral treatment and normal educational and intellectual functioning in young autistic children. *Journal of consulting and clinical psychology*, 55(1):3, 1987.

[12] NLTK. Nltk tokenize package.

[13] Johnny Matson. Aggression and tantrums in children with autism: A review of behavioral treatments and maintaining variables. *Journal of Mental Health Research in Intellectual Disabilities*, 2(3):169–187, 2009.

[14] Cathleen C Piazza, Gregory P Hanley, Lynn G Bowman, John M Ruyter, Steven E Lindauer, and Deborah M Saiontz. Functional analysis and treatment of elopement. *Journal of applied behavior analysis*, 30(4):653–672, 1997.

[15] Joshua L Lipschultz and David A Wilder. Behavioral assessment and treatment of noncompliance: A review of the literature. *Education and Treatment of Children*, 40(2):263–298, 2017.

[16] Edward G Carr. The motivation of self-injurious behavior: a review of some hypotheses. *Psychological bulletin*, 84(4):800, 1977.

[17] Johnny L Matson and Santino V LoVullo. A review of behavioral treatments for self-injurious behaviors of persons with autism spectrum disorders. *Behavior Modification*, 32(1):61–76, 2008.

[18] American Psychiatric Association et al. *Diagnostic and statistical manual of mental disorders (DSM-5®)*. American Psychiatric Pub, 2013.

[19] John T Rapp and Timothy R Vollmer. Stereotypy i: A review of behavioral assessment and treatment. *Research in developmental disabilities*, 26(6):527–547, 2005.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[21] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[23] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014.

[24] Gensim. Word2vec embeddings.

[25] SciKit Learn. Tf-idf vectorizer.

[26] Gensim. Doc2vec paragraph embeddings.

[27] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[28] SciKit Learn. C-support vector classification.

[29] SciKit Learn. Gaussian process classification (gpc) based on laplace approximation.

[30] Autism Speaks. CDC update on autism shows gap between early concerns and evaluation. `https://www.autismspeaks.org/news/news-item/cdc-update-autism-shows-gap-between-early-concerns-and-evaluation`. Accessed: 2016-05-14.

[31] Louis P Hagopian, David M Wilson, and David A Wilder. Assessment and treatment of problem behavior maintained by escape from attention and access to tangible items. *Journal of Applied Behavior Analysis*, 34(2):229–232, 2001.

[32] Johnny L Matson, Dennis R Dixon, and Michael L Matson. Assessing and treating aggression in children and adolescents with developmental disabilities: a 20-year overview. *Educational Psychology*, 25(2-3):151–181, 2005.

[33] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[34] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[35] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[36] Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 111–121, 2014.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[38] Bahar Salehi, Paul Cook, and Timothy Baldwin. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 977–983, 2015.

[39] Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. *arXiv preprint arXiv:1509.01692*, 2015.

[40] Lei Chen, Gary Feng, Chee Wee Leong, Blair Lehman, Michelle Martin-Raugh, Harrison Kell, Chong Min Lee, and Su-Youn Yoon. Automated scoring of interview videos using doc2vec multimodal feature extraction paradigm. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ICMI '16, pages 161–168, New York, NY, USA, 2016. ACM.

[41] Ilia Markov, Helena Gómez-Adorno, Juan-Pablo Posadas-Durán, Grigori Sidorov, and Alexander Gelbukh. Author profiling with doc2vec neural network-based document embeddings. In *Mexican International Conference on Artificial Intelligence*, pages 117–131. Springer, 2016.

[42] Sangheon Lee, Xiangdan Jin, and Wooju Kim. Sentiment classification for unlabeled dataset using doc2vec with jst. In *Proceedings of the 18th Annual International Conference on Electronic Commerce: E-Commerce in Smart Connected World*, ICEC '16, pages 28:1–28:5, New York, NY, USA, 2016. ACM.

[43] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[44] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.

[45] JA Kosmicki, V Sochat, M Duda, and DP Wall. Searching for a minimal set of behaviors for autism detection through feature selection-based machine learning. *Translational psychiatry*, 5(2):e514, 2015.

[46] Dennis Paul Wall, J Kosmicki, TF Deluca, E Harstad, and Vincent Alfred Fusaro. Use of machine learning to shorten observation-based screening and diagnosis of autism. *Translational psychiatry*, 2(4):e100, 2012.

[47] Changchun Liu, Karla Conn, Nilanjan Sarkar, and Wendy Stone. Physiology-based affect recognition for computer-assisted intervention of children with autism spectrum disorder. *International journal of human-computer studies*, 66(9):662–677, 2008.

[48] Erik Linstead, Rene German, Dennis Dixon, Doreen Granpeesheh, Marlena Novack, and Alva Powell. An application of neural networks to predicting mastery of learning outcomes in the treatment of autism spectrum disorder. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 414–418. IEEE, 2015.

[49] Erik Linstead, Dennis R Dixon, Ryan French, Doreen Granpeesheh, Hilary Adams, Rene German, Alva Powell, Elizabeth Stevens, Jonathan Tarbox, and Julie Kornack. Intensity and learning outcomes in the treatment of children with autism spectrum disorder. *Behavior modification*, 41(2):229–252, 2017.

[50] Matthew J Maenner, Marshalyn Yeargin-Allsopp, Kim Van Naarden Braun, Deborah L Christensen, and Laura A Schieve. Development of a machine learning algorithm for the surveillance of autism spectrum disorder. *PloS one*, 11(12):e0168224, 2016.

[51] Jina Jang, Dennis R Dixon, Jonathan Tarbox, and Doreen Granpeesheh. Symptom severity and challenging behavior in children with asd. *Research in Autism Spectrum Disorders*, 5(3):1028–1032, 2011.

[52] Elizabeth Stevens, Abigail Atchison, Laura Stevens, Esther Hong, Doreen Granpeesheh, Dennis Dixon, and Erik Linstead. A cluster analysis of challenging behaviors in autism spectrum disorder. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 661–666. IEEE, 2017.

[53] K McClintock, S Hall, and Chris Oliver. Risk markers associated with challenging behaviours in people with intellectual disabilities: a meta-analytic study. *Journal of Intellectual Disability Research*, 47(6):405–416, 2003.

[54] Jordan Ott, Abigail Atchison, and Erik J Linstead. Exploring the applicability of low-shot learning in mining software repositories. *Journal of Big Data*, 6(1):35, 2019.

[55] Johnny L Matson and Michael L Matson. *Comorbid conditions in individuals with intellectual disabilities.* Springer, 2015.

Appendix A

Source Code Samples

```
# CARD Skills Dataset
# Behavioral Descriptions and Labels

## ---------------------------------------------------------------
# Preproccessing and tokenization

import pandas as pd
from nltk.tokenize import TweetTokenizer

#Script to strip all non alphabetic (or space) characters
#and tokenize the result
def tokenize(passage):
    tokenizer = TweetTokenizer(preserve_case=False)
    try:
        passage = re.sub('[^a-zA-Z ]+', '', passage)
        passage = tokenizer.tokenize(passage)
        print(passage)
        return passage
    except:
        return 'NC'

#Script to tokenize a pandas dataframe with definition column
def process(data):
    data['tokens'] = data['definition'].progress_map(tokenize)

    #remove all data points for which tokenization failed
    data = data[data.tokens != 'NC']
    data.reset_index(inplace=True)

    return data

data = pd.read_csv('Document location of csv file
organized as categoryName,definition')
data = process(data)
```

Figure A.1: Processing and tokenization of behavioral descriptions

```
## ----------------------------------------------------------------
# Word2Vec Generation

from gensim.models.word2vec import Word2Vec

#Keep only the 7 categories we will be studying
categoriesToInclude = ["Aggression","Disruption",
                       "Noncompliance","Stereotypy",
                       "Tantrums","Self-Injurious Behavior",
                       "Elopement"]
df = data.loc[(data['categoryName'].isin(categoriesToInclude))]

#Build Word2Vec Model
w2vModel = Word2Vec(size=50, min_count=5, sg=1, workers=5)
w2vModel.build_vocab(df['tokens'].tolist())
w2vModel.train(df['tokens'].tolist(),
               total_examples=w2vModel.corpus_count,
               epochs=w2vModel.iter)

#Find the most similar vectors to the vector representing word
word = "some string we want to analyze"
w2vModel.wv.most_similar(word)
```

Figure A.2: Word2Vec Generation

```
## --------------------------------------------------------------
# TF-IDF Score calculation from W2V model created above

from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

vectorizer = TfidfVectorizer()
response = vectorizer.fit_transform(" ".join(obj)
                                    for obj in df['tokens'])
index_value={i[1]:i[0] for i in vectorizer.vocabulary_.items()}
tfidfTokens = []
for row in response:
    tfidfTokens.append({index_value[column]:value
                        for (column,value) in
                        zip(row.indices,row.data)})

tfidfs = np.asarray(tfidfTokens)
df['tfIdf'] = tfidfs




## --------------------------------------------------------------
# Weighted W2V generation from W2V model and TF-IDF scores
weightedW2VDocumentVectors = []
sizeOfW2V = 50

for document in df['tfIdf']:
    currentVector = np.zeros(sizeOfW2V)
    for word, tfidf in document.items():
        try:
            currentVector += w2vModel[word] * tfidf
        except:
            print(word,"not included in model")
    #document vector has been built
    weightedW2VDocumentVectors.append(currentVector)

df['weightedWord2Vec'] = weightedW2VDocumentVectors
#saving dataframe
df.to_pickle("ASD_Behaviors")
```

Figure A.3: Weighted Word2Vec Document Generation

```
## ------------------------------------------------------------
# Doc2Vec generation from dataframe created above

import pandas as pd

#preproc
from sklearn import datasets
from sklearn import preprocessing
from sklearn.model_selection import StratifiedShuffleSplit
from gensim.models.doc2vec import TaggedDocument

data = pd.read_pickle("ASD_Behaviors")
X = data['tokens']
Y = data['categoryName']

sss = StratifiedShuffleSplit(n_splits = 1, test_size=0.2,
                             random_state = 0)
train_index, test_index = sss.split(X,Y):

test = data.iloc[test_index]
train = data.iloc[train_index]

#Visualize the label splits
print(y_train.value_counts())
print(y_test.value_counts())

#Build a dictionary of the number labels for all classes
tagsAsNumbers = {}
num = 0
for tag in sorted(y_train.unique()):
    tagsAsNumbers[tag] = num
    num += 1

trainDocs = [TaggedDocument(words, [tagsAsNumbers[tag]])
             for words, tag in
             zip(train['tokens'],train['categoryName'])]

testDocs = [TaggedDocument(words, [tagsAsNumbers[tag]])
             for words, tag in
             zip(test['tokens'],test['categoryName'])]
```

```
#doc2vec
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
import multiprocessing

cores = multiprocessing.cpu_count()

#dm ({0,1}, optional) 1: PV-DM, 0: PV-DBOW
#vector_size how big the paragraph vectors will be
#hs ({1,0}, optional)  If 1, hierarchical softmax will be used
#                      If set to 0, and negative is non-zero,
#                      negative sampling will be used.
#negative - if negative sampling, how many noise words to use

d2vModel = Doc2Vec(dm=0,
                       vector_size=50,
                       hs=0, negative=5,
                       min_count=2,
                       sample = 0,
                       workers = cores,
                       alpha=0.025,
                       min_alpha=0.001)

#initialize model weights
d2vModel.build_vocab([doc for doc in trainDocs])

d2vModel.train(trainDocs,
               total_examples=len(trainDocs), epochs=30)

#Given a Doc2Vec model and a list of TaggedDocuments as input
#vectorLearning returns y(targets), x(inputs) to be used in ML
def vectorLearning(model, inputDocs):
    targets, vectors = zip(*[(doc.tags[0],
                              model.infer_vector(doc.words,
                                                 epochs=20))
                       for doc in inputDocs])
    return targets, vectors

#example
ytrain, xtrain = vectorLearning(d2vModel, trainDocs)
ytest, xtest = vectorLearning(d2vModel, testDocs)
```

Figure A.4: Doc2Vec Generation

```
## ---------------------------------------------------------------
####SVM classification on Document Vectors

from sklearn.svm import SVC
import numpy as np

def runSVM(XTrain, yTrain, XTest, yTest, title=None):

    #Build SVM on training set
    svm = SVC()
    svm.fit(XTrain, yTrain)

    #Generate predictions
    yTrainPred = svm.predict(XTrain)
    yTestPred = svm.predict(XTest)

    if title:
        print(title)

    #Output confusion matrix
    print(metrics.confusion_matrix(yTest, yTestPred))

    #Output testing and training accuracies
    trainAccuracy = np.mean(np.asarray(yTrainPred).ravel() ==
                            np.asarray(yTrain).ravel()) * 100
    testAccuracy = np.mean(np.asarray(yTestPred).ravel() ==
                           np.asarray(yTest).ravel()) * 100
    print('Train accuracy: %.1f' % trainAccuracy)
    print('Test accuracy: %.1f' % testAccuracy)


#example on train and test pandas df
#with 2 class labels in 'Label' and document vectors in 'D2V'
yTrain = train['Label'].to_list()
yTest = test['Label'].to_list()

XTrain = train['D2V'].to_list()
XTest = test['D2V'].to_list()

runSVM(XTrain, yTrain, XTest, yTest)
```

Figure A.5: SVM Training and Testing

```
## ------------------------------------------------------------
####GPC on Document Vectors

#Gaussian
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn import metrics

#Default kernel is RBF, explicitly stated here
kernel = 1 * RBF()
gpc = GaussianProcessClassifier(kernel = kernel,
                                random_state = 2)

#Fit GPC to a dataframe with document vectors column 'D2VVectors'
#and numeric behavioral labels in column 'Label'
gpc.fit(train['D2VVectors'].to_list(),train['Label'].to_list())

##Prediction
yPredTrain = gp_neg.predict(train['D2VVectors'].to_list())
yPredTest = gp_neg.predict(test['D2VVectors'].to_list())

trainAccuracy = np.mean(np.asarray(yPredTrain).ravel() ==
                        np.asarray(train['Label'].to_list()).ravel())

print('Train accuracy: %.1f' % trainAccuracy*100)

testAccuracy = np.mean(np.asarray(yPredTest).ravel() ==
                       np.asarray(test['Label'].to_list()).ravel())
print('Test accuracy: %.1f' % testAccuracy*100)

#Generate confusion matrix
metrics.confusion_matrix(test['Label'].to_list(), yPredTest)
```

Figure A.6: Gaussian Process Classification

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import matplotlib

#Given a pandas dataframe, dataSet, with a
#column of document vectors called 'D2V' and a
#column of corresponding labels called 'categoryName'
#Generate a scatter plot and save at the file location saveAt
def plotDocVecs2(dataSet,saveAt):

    labels = list(dataSet['categoryName'])
    vectors = dataSet['D2V'].as_matrix().tolist()
    uniqueLabels = list(set(labels))

    #Reduce vectors down to a vector space of 2
    tsne_model = TSNE(perplexity=40, n_components=2, init='pca',
                      n_iter=2500, random_state=23)
    new_values = tsne_model.fit_transform(vectors)
    print("Data reduction complete")
    x = [value[0] for value in new_values]
    y = [value[1] for value in new_values]

    plt.figure(figsize=(16,16))
    #Plot each point as either class 1 or class 2
    for i in range(len(x)):
        if labels[i] == uniqueLabels[0]:
            plt.scatter(x[i],y[i],c="red",marker="o")
        elif labels[i] == uniqueLabels[1]:
            plt.scatter(x[i],y[i],c="black",marker="x")

    #Generate legend (not used in visualizations in this paper)
    redCircle = mlines.Line2D([],[],color="red",marker="o",
                  linestyle = "None",label=uniqueLabels[0])
    blueSquare = mlines.Line2D([],[],color="black",marker="x",
                  linestyle = "None",label=uniqueLabels[1])
    plt.legend(handles=[redCircle,blueSquare],fontsize='xx-large')

    #Save the figure as an eps file
    plt.savefig(saveAt, format='eps', dpi=500,bbox_inches = 'tight')
    #Display figure
    plt.show()
```

Figure A.7: T-SNE Visualization