signals

MDPI

Article

Dialogue Act Classification via Transfer Learning for Automated Labeling of Interviewee Responses in Virtual Reality Job Interview Training Platforms for Autistic Individuals

Deeksha Adiani [1,*], Kelley Colopietro [2], Joshua Wade [2], Miroslava Migovich [2], Timothy J. Vogus [3] and Nilanjan Sarkar [1,2]

[1] Computer Science, Vanderbilt University, Nashville, TN 37212, USA; nilanjan.sarkar@vanderbilt.edu
[2] Mechanical Engineering, Vanderbilt University, Nashville, TN 37212, USA; kelley.j.colopietro@vanderbilt.edu (K.C.); miroslava.migovich@vanderbilt.edu (M.M.)
[3] Owen Graduate School of Management, Vanderbilt University, Nashville, TN 37203, USA; timothy.j.vogus@vanderbilt.edu
* Correspondence: deeksha.m.adiani@vanderbilt.edu

Abstract: Computer-based job interview training, including virtual reality (VR) simulations, have gained popularity in recent years to support and aid autistic individuals, who face significant challenges and barriers in finding and maintaining employment. Although popular, these training systems often fail to resemble the complexity and dynamism of the employment interview, as the dialogue management for the virtual conversation agent either relies on choosing from a menu of prespecified answers, or dialogue processing is based on keyword extraction from the transcribed speech of the interviewee, which depends on the interview script. We address this limitation through automated dialogue act classification via transfer learning. This allows for recognizing intent from user speech, independent of the domain of the interview. We also redress the lack of training data for a domain general job interview dialogue act classifier by providing an original dataset with responses to interview questions within a virtual job interview platform from 22 autistic participants. Participants' responses to a customized interview script were transcribed to text and annotated according to a custom 13-class dialogue act scheme. The best classifier was a fine-tuned bidirectional encoder representations from transformers (BERT) model, with an f1-score of 87%.

Keywords: dialogue act classification; transfer learning; autism spectrum disorder; virtual reality job interviews

Citation: Adiani, D.; Colopietro, K.; Wade, J.; Migovich, M.; Vogus, T.J.; Sarkar, N. Dialogue Act Classification via Transfer Learning for Automated Labeling of Interviewee Responses in Virtual Reality Job Interview Training Platforms for Autistic Individuals. *Signals* **2023**, *4*, 359–380. https://doi.org/10.3390/signals4020019

Academic Editors: Manuel Duarte Ortigueira and Vessela Krasteva

Received: 14 January 2023
Revised: 26 April 2023
Accepted: 11 May 2023
Published: 19 May 2023

1. Introduction

Autism spectrum disorder (ASD) is a lifelong neurological and developmental disorder, with a prevalence of 1 in 44 children in the United States (US) [1], which affects an individual's ability to learn, as well as to communicate and interact socially. ASD is characterized by deficits and/or impairments in social and emotional reciprocity, in communicative nonverbal behaviors, and in forming and maintaining relationships; the level of severity varies, where an individual may require minimal to substantial support [2]. Autistic individuals (identity-first terminology has been chosen based on preference of autistic individuals [3]) are often under- or unemployed, and finding and keeping employment is often difficult [4–6]. A 2017 National Autism Indicators Report indicated that amongst 3520 autistic adults (ages 18–64) from 31 states in the US who received developmental disability services, only 14% were employed with pay in the community, 54% had unpaid jobs or activities in a facility, and 27% had no jobs or any form of participant in the community [4]. Some factors for these low rates of employment amongst autistic individuals include difficulties in social interaction, discrimination in the workplace [7], sensitivity

to workplace environments (e.g., lighting, sounds, smells, etc.) [8], lack of support from vocational rehabilitation services or a lack of services thereof [9], and/or a poor job fit [8]. Amongst those challenges, the job interview, where autistic individuals are expected to adhere to the "norm" in terms of mannerisms and interactions, poses a major barrier to employment, and in turn, to independent living [7,10,11].

To address the above, several computer-based job interview simulators have been developed, catering to autistic individuals' needs (e.g., routine and certainty), which have demonstrated efficacy [12–17]. VR-based interview training systems, including VR simulations, provide autistic job candidates a controlled self-paced learning environment to practice their interviewing skills across multiple scenarios with lower interpersonal risk and a sense of psychological safety [12,18,19]. Studies have shown that technology-enhanced job interview training platforms can improve interviewing skills and overall self-confidence when compared to traditional interview training methods [12,20]. Further, virtual job interview training can potentially reduce costs, as it reduces the need for personnel to conduct real-life mock interviews [21]. Studies have demonstrated that depending on its realism, a virtual interview can induce experiences of anxiety or nervousness consistent with real interviews, which can help candidates get accustomed to the affective experience of interviews, and in turn, feel more comfortable during a real interview scenario [22–24]. Previous virtual job interview simulations with conversation agents, aimed at helping autistic individuals, collect interviewee (participant) responses and perform keyword extraction on transcribed text, which is dependent on the domain of the interview script [12,14–16,19]; oftentimes, the interview systems do not allow for freely spoken responses, and instead provide multiple answers from which to choose the appropriate one. Though the existing methods work, they are ad hoc solutions, and the interview systems lack text classification methods which can recognize the intent of the interviewee's spoken utterances, independent of the domain of the interview script. In our recent work [25], we show how manually identifying intents from autistic interviewees' utterances can provide insight into their performance. Hence, automatic labeling to recognize the meaning of utterances has the potential to enhance current simulators for training autistic individuals. We address this via automated dialogue act classification.

Dialogue act classification or recognition is a method where utterances of a conversation are classified as one of several dialogue acts or speech acts [26]. A dialogue act (DA) or speech act represents the meaning or intent of an utterance, e.g., whether the utterance is a question, a yes or no response, a "thank you" or a "good-bye", or a statement of opinion or fact [27]. Hence, DAs can capture the semantics of utterances in a domain by mapping utterances to DA. DA classification has several applications, such as intent mining and identification [28], classifying issues from comments on online collaborative platforms such as GitHub [29] in conversation agents and dialogue systems for an agent/system to understand the user and execute tasks that drive the flow of dialogue [30–33]. When combined with conversation context, DA classification has shown to improve contextual topic modeling (extraction of themes or topics in text [34]) [35]. When automated, DA classification can be used to automatically label utterances, whether from text or other modes of data, such as text combined with facial expressions [36,37] or speech signals [32,38].

During the development of our previous work [39], we encountered a lack of publicly available data to train a text classifier that classifies utterances into domain-independent DA that are relevant in a job interview context. Hence, in response to the aforementioned gap, we collected utterances of participants responding to questions from an interview script while interacting with our virtual job interview platform described in [39]. This data collection study resulted in a small dataset of 640 utterances. Observing patterns in the responses to interview questions, we developed a DA labeling scheme based on previous works [32,40,41] that captures the semantics of utterances in the context of a job interview regardless of the domain of the interview. The anonymized data were then labeled by two annotators according to the developed DA scheme. These annotated data were then used to train the classifier. First, a baseline support vector machine (SVM) classifier and a random

forest (RF) classifier were trained on the annotated data, which helped understand our data and highlighted where traditional machine learning failed to classify some labels. Second, a pretrained bidirectional encoder representations from transformers (BERT) model was fine-tuned using transfer learning on our annotated data (building on the work of [41]). The final BERT classifier performed with a 0.92 accuracy and an f1-score of 0.87.

The scope of this article is twofold: (1) to introduce a DA scheme to capture semantics of utterances in a job interview context through data collected in an experimental study with autistic individuals using our virtual reality job interview training system [39]; and (2) to present the best-performing classifier on the data. The three main contributions of this paper are (1) an original dataset comprising annotated responses to job interview questions by autistic individuals of working age, where responses are labeled using an in-house dialogue act scheme for job interviews; (2) a predictive model that takes an interview response as input and outputs the dialogue act to understand the general intent of the response; and (3) the interview script and annotation guidelines to help future developers replicate this work and collect more data. In this work, we aim to introduce a new autistic dataset with real speech data collected from a simulated job interview that can be used for automated DA classification in VR simulations to aid autistic individuals. It is to be noted that we have adapted a standard BERT model for DA classification of utterances to demonstrate feasibility as a first step towards this aim. More sophisticated models and/or methods can be employed in the future to better the accuracy.

2. Related Work

Previous studies discuss several methods for automated DA classification. Early DA classification models were traditional machine learning (ML) methods such as support vector machines [42,43], sometimes combined with hidden Markov models [44], Bayesian network-based classifiers [45,46], decision trees [47], and more [48]. Recently, solutions for DA classification involve methods in deep learning. Khatri et al. [35] created a 14-DA scheme based on observed patterns in a user–chatbot interaction dataset to determine the goal from the user's utterances. Examples of DAs were InformationRequest, InformationDelivery, GeneralChat, TopicSwitch, and more. They used DA as context to improve the accuracy of a topic modeling algorithm, where the best performing DA classifier was a contextual deep average network (DAN) model. Raheja and Tetreault [49] combined a context-aware self-attention mechanism with a hierarchical recurrent neural network (RNN) to conduct utterance-level DA classification. They demonstrated improvements in classification on the Switchboard Dialogue Act (SwDA) Corpus [50] , which is a benchmark dataset with a 42 DA scheme to capture general conversation [40]. Ahmadvand et al. [31] introduced a novel method for contextual DA classification built upon the method of [35]. The features included utterances represented as 300-dimension Word2Vec embeddings, parts-of-speech (POS) tags, topics, and lexical features, including word count and sentence count. Their model was a fully connected convolutional neural network tested on known state-of-the-art conversation corpora, demonstrating improved accuracy relative to previously reported baselines. Chatterjee and Sengupta [28] introduced an extension to the density-based algorithm (DBSCAN) [51] for intent identification for conversation agents by using available conversation data. Their motivation was to help reduce the need for labeling conversation data by automatically labeling previously available data with their classifier. The results on six datasets show the potential of their algorithm in intent mining. Although the above have shown promise, they do not show improvements as significant as those demonstrated via transfer learning [52] using pretrained BERT [53]. Transfer learning in ML is an approach that involves improved learning in a new task via transfer of knowledge from a previously learned related task, where the previously trained model is further trained or "fine-tuned" on new task data [54].

Duran et al. [55] conducted experiments to investigate the results using different single-sentence representations in training DA classifiers, which included embeddings, punctuation, text case, vocabulary size, and tokenization. The results demonstrated the

impact of the different features in sentence representations for the training of several models. The results also implied that the pretrained BERT and a variation of it called RoBERTa [56] showed notable increase in accuracy on the SwDA corpus when compared to other models, which demonstrated that contextual sentence representations produced by the BERT models are an improvement over other fine-tuned models. In another study, Noble and Maraev [57] found that while a standard pretrained BERT performs well, for good performance on task-specific DA classification, fine-tuning the pretrained BERT is essential. Wu et al. [58] further pretrained a BERT model on eight datasets using masked language modeling (MLM), which is a fill-in-the-blank pretraining strategy where the model is trained to replace a [MASK] token in a sentence with a random input token (e.g., "Paris is the [MASK] of France", where [MASK] can be replaced by the token which contextually makes sense such as "capital"). Their task-oriented dialogue-BERT, or TOD-BERT, can be further fine-tuned for tasks such as response selection, dialogue state tracking, intent recognition, and DA prediction, which can help address the task-oriented dialogue data scarcity problem. Chakravarty et al. [41] created a DA scheme to fit the context of a question-answering scenario where DAs were developed for questions and answers, respectively (e.g., wh-question or yes-answer) . They trained three models on an annotated question-answering dataset with transcribed text from interviews: a convolutional neural network (CNN), a Bi-LSTM, and a BERT-base-cased pretrained model. Amongst the three fine-tuned models, their BERT model performed the best, with an overall F1-score of 0.84.

From the above literature review, we concluded that fine-tuning a pretrained model via transfer learning demonstrates significantly better results than training a model "from scratch". We can also see in previous literature [41,58] that the pretrained BERT models perform significantly better when it comes to text classification, which includes DA classification. Hence, we chose to train a DA classifier using a transfer learning approach.

3. Materials and Methods

3.1. Data Collection via Job Interview Training System

A total of 22 autistic participants (mean age = 19.1; SD = 2.8; 17 male and 5 female; 20 White—of which 4 were Hispanic/Latino—and 2 African American) were recruited for a data collection study. Participants were required to be verbal, fluent in English, and at the eligible age of employment (16+ years) in the state of Tennessee, and the study was approved by the Institutional Review Board (IRB) of the lead author's university. Participants went through a mock interview with our career interview readiness in virtual reality (CIRVR) job interview environment with a virtual interviewer avatar [39]. During the interview simulation, the participant is seated across a desktop computer that runs the CIRVR simulation, as shown in Figure 1, where the participant can interact with CIRVR and move around the virtual space using mouse and keyboard controls. Spoken responses are captured via a microphone headset, which are then submitted via keyboard controls. This feature allows the user some time to think about their response before submission. Although CIRVR provides an immersive VR option, we chose not to use a VR headset or head-mounted display for this study due to preferences of the population, as mentioned in our previous work [39].

To ensure a realistic job interview experience in CIRVR, the interview script comprised questions for the position of a data entry clerk. The questions reflected both a review of multiple years of questions used by actual employers in a database of a university career management center, and interviews with 36 autistic employees, support professionals (e.g., job coaches and university career center professionals), and employers. The interview structure consisted of the following segments, as described in [15]: greetings ("hello" and "good morning"); technical questions (about work experience and specific job skills); education questions (e.g., formal schooling, favorite subjects, cocurricular activities, or job-relevant hobbies and activities); personal questions (e.g., behavior-based questions, such as experience working in a team); and an interviewee-initiated questions section, where the participant can ask questions about the job (e.g., the work environment and

work hours). CIRVR went through interview questions from a predefined interview script, which the virtual avatar output as speech using Microsoft Azure Text-to-Speech synthesizer. The participants' responses to each question were captured via a microphone on a headset and were transcribed in real-time using Microsoft Azure Speech-to-Text [59], which has been benchmarked as the speech recognizer with lowest transcription error rates when compared with competitors [60]. Transcribed speech for each session was stored as text along with the corresponding questions, in a comma-separated values (CSV) file. These files were then analyzed to create a DA labeling scheme, as discussed in the next subsection.

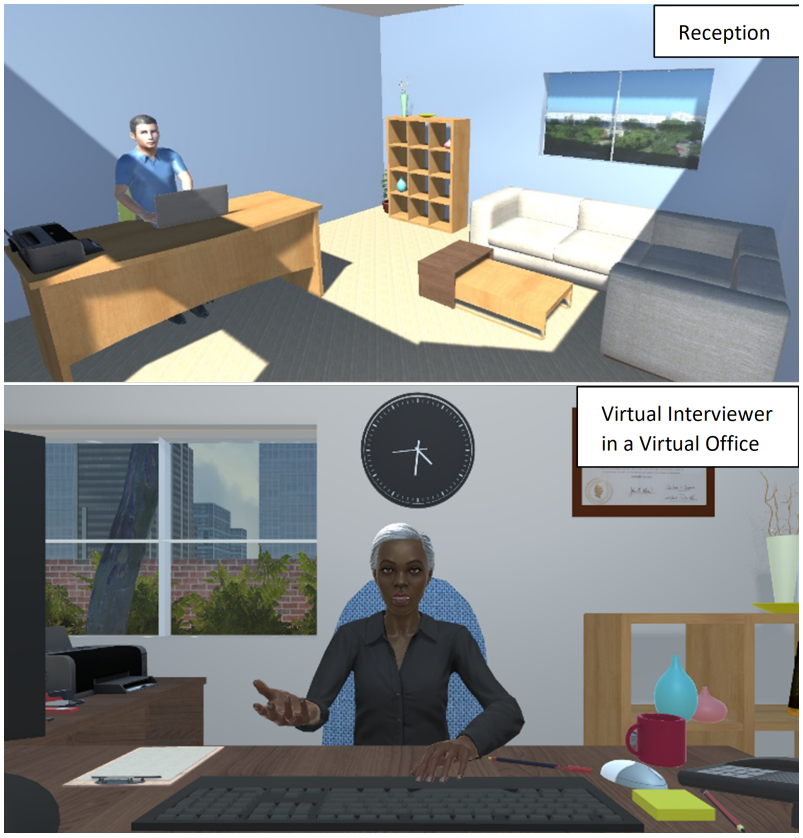

Figure 1. The CIRVR virtual environment: the reception (**top**); the virtual avatar behind a desk in a virtual office from the perspective of the interviewee (**bottom**).

3.2. The BERT Architecture and Fine-Tuning BERT for Text Classification

BERT is a transformer-based model, first introduced by Devlin et al. [53], that has been pretrained via self-supervised learning on a large corpus of unlabeled data in English. It consists of a multilayer transformer encoder architecture that is based on the original transformer encoder, as described in [61]. Figure 2 presents a visualization of the BERT *base* model that has 12 encoder layers, a hidden size of 768, and 12 attention heads [53]. Each encoder consists of (1) a self-attention layer, where a sentence is passed to help the model learn to which words it should "pay attention" to based on context of the surrounding words; and (2) a feedforward neural network layer that processes the output from the attention layer to format/fit it into an acceptable form for the next encoder [53,61]. Pretraining of BERT was conducted over two tasks: MLM and next sentence prediction (NSP). In MLM, a model is trained to predict missing words in a sentence based on the context provided by neighboring words in a sentence. In the MLM task for BERT, 15% of the words in an input sentence were masked, and the masked sentence was run though the BERT model for prediction of the missing words. This was carried out to train a model with the ability to learn a deep bidirectional representation of the sentence. During the NSP task, the input consisted of two sentences in order to train a model to understand the relationship between them, i.e., is sentence B next to sentence A or not [53]. Pretraining of

BERT using these two methods prepares the model for downstream tasks such as question answering, or in our case, sentence classification; specifically, single-sentence classification.

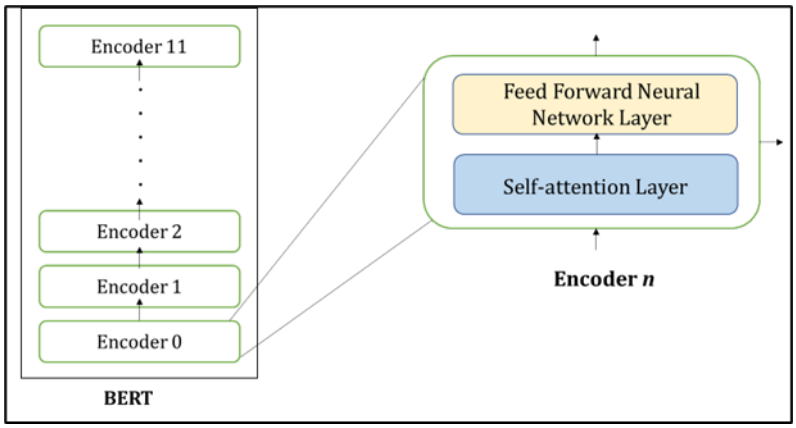

Figure 2. The BERT architecture, consisting of stacked encoders; each encoder consists of a self-attention layer followed by a feedforward neural network layer.

To understand how BERT classifies text, let us take an example such as "I like cooking". This sentence is first converted to lowercase (we chose the BERT-base-uncased model for fine tuning). Then, the sentence is tokenized as per the BERT tokenizer, which gives the Tokenized Input, as shown in Figure 3. The word "cooking" is stemmed or broken into two words, "cook" and "##ing", so as to not increase the size of the vocabulary, as well as to be able to handle any variation of the word "cook" such as "cooks", which will be broken into "cook" and "##s". Each token has a pretrained embedding, and collectively for a sentence input, they form the Token Embeddings. The [CLS] token is a special classification task token, and the [SEP] token is a separator between multiple sentences; so, there can be a sentence after [SEP] for multi-sentence classification. The next layer of inputs are the learned embeddings that state the sentence to which the token belongs. In our example, we have a single sentence only, so all will be E_A, i.e., each token belongs to sentence A. The final inputs are the Positional Embeddings, which index the position of the tokens in the vocabulary. The sum of the three embeddings is the input to the BERT encoder. Each token, when passed through the encoders, is embedded into a vector of length 768. In our example in Figure 3, the 4 tokens in a sentence plus the 2 special tokens form an embedding matrix of size 6×768, where 768 corresponds to the number of hidden units. This is the Model Output. For the text classification task, BERT takes the final hidden state of the [CLS] token, which is a representation of the whole sequence (the input sentence), and passes it to the Classifier to predict the label. The Classifier here can be a feedforward neural network on whose output we apply SoftMax to obtain the probabilities for each class. The class with the maximum probability value is the predicted label [62].

3.3. Dialogue Act Labeling

The DAs to label the interview questions were adapted from the Questions DA scheme in [41], with modifications made to the descriptions and guidelines to fit the context of our data, and in general, for the job interview context. These DAs and our modified descriptions of the data are given in Table 1. The interviewees' utterances/responses were classified according to the scheme in Table 2, which is an adaptation of the Answers DA scheme from [41], with the addition of six labels to capture all types of utterances in our dataset: *xx*, *query*, *ft*, *fa*, *fp*, *fe*. The *xx* label was used to account for any responses that were ambiguous or uninterpretable, and the *query* label was used to account for interviewee-initiated questions. The last four of the six labels were adapted from [32]. Based on patterns observed in the mock interview's text, a set of in-house guidelines were developed for annotating the questions and the participant utterances according to the DA labeling schemes in Tables 1 and 2.

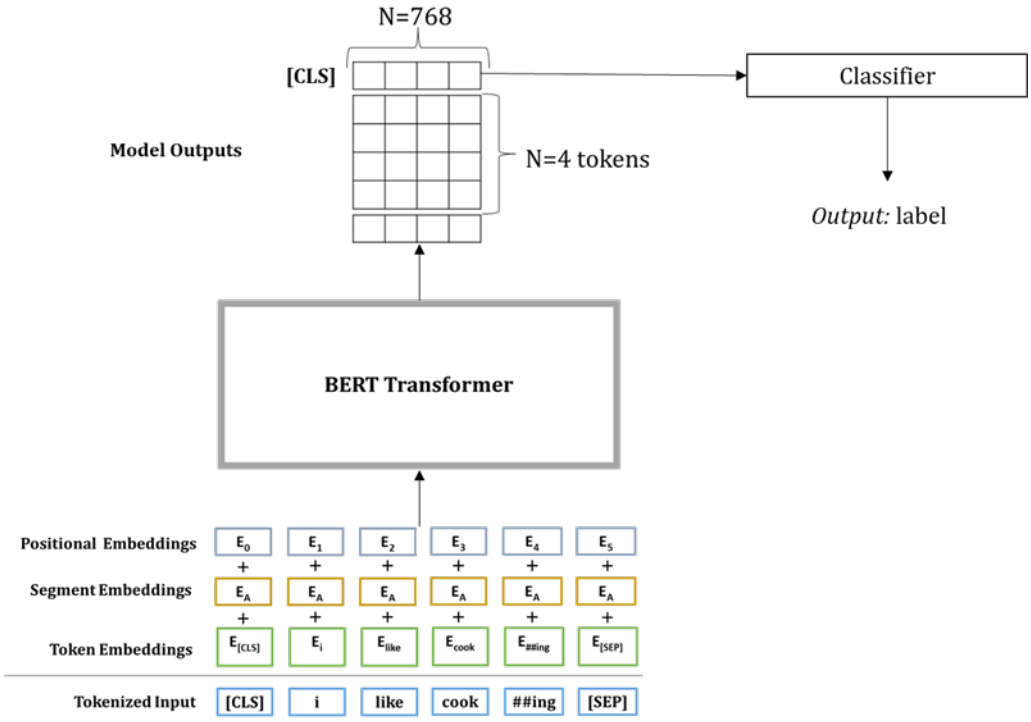

Figure 3. Fine-tuning BERT for text classification, as described in [53].

Table 1. Questions Dialogue Acts.

Dialogue Act	Description	Example
wh	Wh-questions, which generally start with "who", "what", "where", "when", "why", "how", etc.	What is your name?
wh-d	Wh-declarative questions, when there is more than one statement in a wh-*question.	You said math is your favorite subject. What kind of grades did you get in Math?
bin	Binary question, which can be answered with a "yes" or "no".	Does that sound good?
bin-d	Binary-declarative question, which can also be answered with a "yes" or "no", but has an explanation or a statement before it.	Before getting into some technical questions about the position, tell me, do you have any prior work experience?
qo	Open question or general questions, not specific to any context, to know the options of the person who is answering.	How do you feel this interview is going?
or	Choice question, made of two parts connected by conjunction "or".	Do you have any experience with spreadsheet software such as Microsoft Excel or Google Spreadsheets?

Since the mock interview questions were the same for all the participants, two annotators first labeled the questions according to the scheme in Table 1. Annotators settled any differences in labels via discussions. The DA labels of the questions were then used as a reference to annotate each participant's/interviewee's response to those questions according to the scheme in Table 2. Types of questions often prompt certain responses [63], e.g., a yes-or-no type of question will often prompt a yes or no type of answer, which can make the annotation less ambiguous and can help increase interannotator agreement. The interannotator agreement on the interviewee utterances, after settling differences, was calculated to be 86%, which was the percentage of utterances labeled with same DA by both annotators out of the total number of utterances. The utterances that the two annotators did not agree upon were discarded, and the ones labeled *xx* were also discarded, as these were uninterpretable and would be noise in the dataset. Finally, we were left with a dataset

of 640 annotated interviewee utterances, and the distribution of labels for the 13 classes is shown in Table 3. Note that the dataset of the participants has been deidentified, and contains placeholders '{personName}' for where the participants introduce themselves to the virtual interviewer.

Table 2. Answers Dialogue Acts.

Dialogue Act	Description	Example
y	Variations of "yes" answers.	"yes", "yeah", "of course", "definitely is", "that's right", "I am sure", etc.
y-d	Yes-answer with an explanation.	Yes. I have experience with Excel.
n	Variations of "no" answers.	"No, I don't think so", "certainly not", "I am afraid not", "not really", "I don't have experience…", "we don't", etc.
n-d	No-answer with an explanation.	"User didn't respond", "I…".
xx	Uninterpretable responses or any responses which look incomplete, such as one word and the user stopped talking.	How do you feel this interview is going?
sno	Non-opinionated statements.	I started working on a project the other day.
so	Opinionated statements.	Anything which starts with "I think", "I believe", "I feel", etc.
ack	Acknowledgements.	"okay", "uh-huh", "I see", etc.
dno	It is a response given when the person is unsure, doesn't know, or doesn't recall.	"I don't know", "maybe", "I guess", "I suppose", etc.
query	Interviewee-initiated question.	What is the work environment like?
ft	Thank yous	"thanks", "thank you"
fa	Apologies.	"I'm sorry"
fe	Exclamations.	"shoot", "oh goodness", "jeez", etc.
fp	Greetings or conventional openings.	"hello", "nice to meet you"

Table 3. Distribution of annotated dataset (n = 640; 13 classes).

Dialogue Act	No. of Samples	Percentage Distribution (%)
y	50	7.81
y-d	17	2.66
n	49	7.66
n-d	10	1.56
sno	381	59.53
so	30	4.69
ack	13	2.03
dno	44	6.88
query	34	5.31
ft	3	0.47
fa	1	0.16
fe	2	0.31
fp	6	0.94

3.4. The Data

From the distribution in Table 3, we can clearly see that the *sno* (non-opinionated statements) label comprises almost 60% of the dataset. This distribution is similar to the distribution in the SwDA benchmark dataset, where *sno* comprises over 50% of the data [40]. The *ft*, *fa*, *fe*, and *fp* classes have <10 samples each, where *fa* has only 1 sample. This would have been an issue during the training and testing split, where the 1 *fa* sample could end up in either the training set or the test set, and the other set would not have any samples from the *fa* class. Instead of random oversampling (creating copies of the minority samples), which is known to lead to overfitting (model trains well on training data, but performs poorly on unseen or test data [64]) [65], we added some new data to the *ft*, *fp*, *fa*, and *fe* classes. Data for *ft*, *fp*, and *fa* were gathered from the SwDA dataset, as those were

general cases of sorry, greetings, and thank you. After isolating the utterances in those classes from the SwDA corpus and removing duplicates, we ended up with 38 samples of *ft*, 57 samples of *fp*, and 41 samples of *fa*. For the *fe* class, we gathered samples from a corpus developed in our previous work, which included exclamation-type words spelled specifically as captured by the Microsoft Azure Speech-to-Text service. For example, curse words are censored by Azure Speech, with the asterisk '*' character replacing the characters (e.g., ****), and other words are spelled differently, such as "jeez", which is spelled as "geez" when transcribed by Azure. After removing duplicates, we ended up with 51 samples of *fe*. We added these to the original 640 sample dataset, which led to an 827-sample dataset. To prepare the dataset for model training, we removed most punctuation with the exception of the following: '?', which was found in interviewee-initiated queries; '{'and '}', which were used for the placeholder '{personName}'; '*', which was left for the curse words that were censored by Azure's Speech-to-Text transcriber; and " ' ", apostrophes used to preserve " 's " or "I'm" in utterances. The utterances were then converted to lowercase, which made our final dataset.

3.5. Model Training

Model training was conducted on a Microsoft Windows 11 PC with 64 GB RAM and a 16 GB NVIDIA Quadro RTX 5000 GPU. To understand the predictive abilities of our small dataset, we trained two classifiers as baseline models. Baseline models are often used to understand the dataset, and help determine the specific classes with which the model fails that may affect later steps in the project, or whether there are sufficient data for the classes [66]. For example, it would help understand whether 10–15 samples in a class are enough or the dataset needs revisiting. We chose support vector machine (SVM) [67] and random forest [68] as the baseline models as they have previously shown to provide acceptable results in DA classification tasks [32,69,70]. The model training process is summarized in Figure 4.

Since our dataset was small and imbalanced, we chose to perform 5-fold cross validation (CV) to obtain the best-performing model as there were not enough data to apply a 3-part split with a training, a validation, and a testing set [71]. We used the scikit-learn *KFold* CV [72] to conduct the training and testing split and set the *shuffle* parameter in the method to *True* to shuffle the data. For reproducibility, we chose a random seed of 47 as a parameter to obtain the same splits with all 13 classes present in the training and the testing datasets. The distribution of the 5 folds of data are given in Table 4. Note that the 5-fold split of data was conducted on the text version of the data and not the encoded version, because on setting the *random_state* parameter of scikit-learn's *KFold* method, the indices of the encoded tokens of utterances change, which is not ideal, as we want to keep the order of tokens the same in the dataset for BERT. After the 5 folds (groups/splits) of data were obtained, the untokenized, unencoded, text versions were saved to comma-separated values (CSV) files to use later for BERT.

For feature extraction for the traditional ML models, we first accumulated all the unique words in the utterances and created a vocabulary of 1305 words. To account for unknown words or words outside of the vocabulary during classification of unseen, new data, we added a token '<UNK>' to the vocabulary list (which made the length of the vocabulary 1306 words). We then tokenized each utterance using the word tokenizer from the Natural Language Toolkit (NLTK) [73], and then one-hot encoded the utterances, where each feature vector representing the utterance was the length of the vocabulary (see Appendix A.1). We used the label encoder [74] from scikit-learn ML toolkit [75] to encode the labels associated with each utterance.

Hyperparameter tuning was conducted via *Randomized Search*. *Randomized Search* [76] is a method of hyperparameter tuning where random combinations from a fixed grid of values (see Table 5) are tried with the data to obtain the hyperparameter values that produce the best-estimating classifier. Scikit-learn's *RandomizedSearchCV* allows the user to perform hyperparameter tuning while cross-validating the combinations of hyperparameters on

the k-fold splits of data, which makes the process faster. After obtaining the best hyperparameters on the 5 folds of data, we ran 5-fold CV again to obtain the classification results on each fold (train and test set) of data. The classification results for each best-performing classifier and the best parameters obtained from hyperparameter tuning are discussed in Section 4. Figure 4 summarizes the aforementioned preprocessing, hyperparameter tuning, and training process.

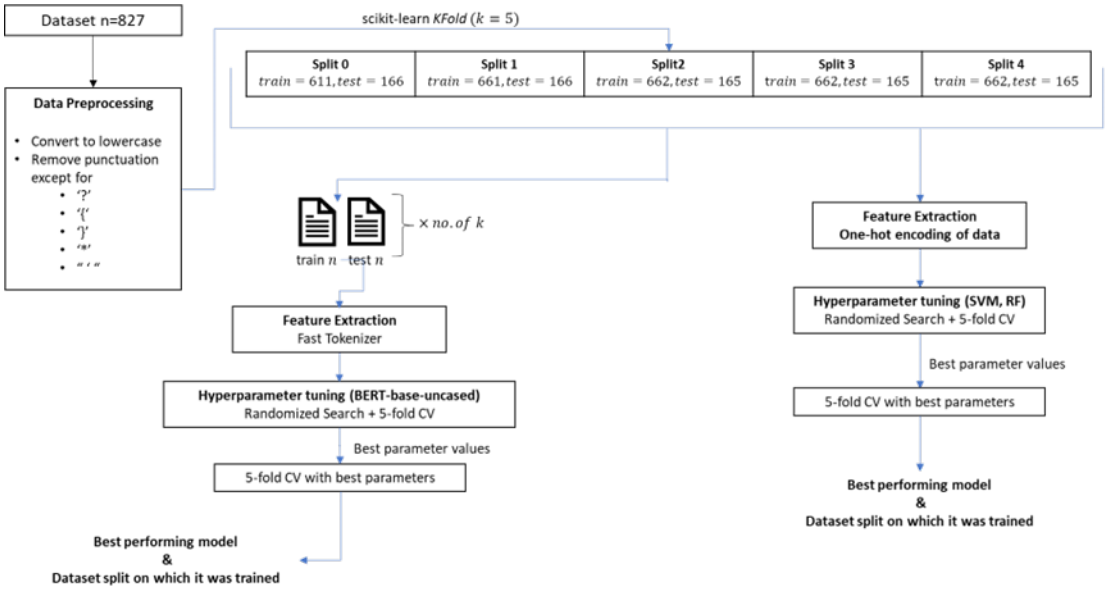

Figure 4. Summary of model training.

Table 4. Distribution of 5 folds (groups) of data (13 classes).

Labels	Split 0		Split 1		Split 2		Split 3		Split 4	
	Train0	Test0	Train1	Test1	Train2	Test2	Train3	Test3	Train4	Test4
y	42	8	42	8	40	10	36	14	40	10
y-d	12	5	12	5	15	2	13	4	16	1
n	36	13	39	10	40	9	42	7	39	10
n-d	8	2	8	2	9	1	7	3	8	2
sno	298	83	300	81	308	73	306	75	312	69
so	25	5	24	6	26	4	23	7	22	8
ack	10	3	12	1	12	1	8	5	10	3
dno	39	5	38	6	30	14	35	9	34	10
query	30	4	29	5	27	7	24	10	26	8
ft	31	10	35	6	33	8	33	8	32	9
fa	36	6	32	10	30	12	37	5	33	9
fe	40	13	42	11	42	11	45	8	43	10
fp	54	9	48	15	50	13	53	10	47	16

Table 5. Hyperparameter grid for each model.

SVM [a]		RF [a]		BERT [b]	
Hyperparameter	**Range of Values**	**Hyperparameter**	**Range of Values**	**Hyperparameter**	**Range of Values**
C	10–100	max_depth	10–110, None	batch_size	8, 16, 32
kernel	radial basis function (rbf) [77], linear	max_features	'auto', 'sqrt'	learning_rate	2×10^{-5}, 3×10^{-5}, 5×10^{-5}
		min_samples_leaf	1, 2, 4	epochs	20, 30, 40
		min_samples_split	1–10		
		n_estimators	10–190		
		bootstrap	True, False		

[a] Hyperparameter names as per scikit-learn documentation. [b] Hyperparameter names as per Hugging Face documentation.

We then moved on to fine-tuning the pretrained BERT model. The data for BERT were utterances that were preprocessed (not encoded), as discussed in Section 3.4, and the labels were encoded by the aforementioned label encoder. The same 5 folds of data that were used for the traditional ML classifiers were also used to cross-validate the BERT model to keep the datasets consistent. The BERT pretrained model and tokenizer were retrieved from the Hugging Face Transformers library [78] ;specifically, we chose the *bert-base-uncased* version on Hugging Face, which consists of 12 transformer encoders stacked together, with a hidden state of 768, 12 attention heads, and 110 million parameters, and ignores case in words (e.g., "Camel" and "camel" are the same). For feature extraction, we used the Fast Tokenizer [79] from Hugging Face. BERT models have variations for several tasks, such as token classification, text classification, language modeling, and question answering. The BERT model for sequence classification has an additional layer for fine-tuning BERT for text classification tasks. All 12 layers of BERT were left unfrozen; hence, the 12 layers were fine-tuned. First, we conducted an in-house randomized search CV over a fixed grid of hyperparameters (see third column of Table 5 on the 5 folds of data. The weight decay was kept constant at 0.01, and the default *AdamW* [80] optimizer was used. On training, we obtain CV scores for each hyperparameter combination on each fold of testing data. The best parameter combination is the one that produced the highest average f1-score over the 5-folds of data. In Section 4, we report the average accuracies and the f1-scores across all 5 folds of data, and highlight the best fine-tuned classifier, as determined by 5-fold CV, with the best parameters found via randomized search CV. In addition, we also report which split of data gave the best classifier. The final predictive BERT classifier was trained on the entire dataset, with the best hyperparameters from the randomized Search CV.

4. Results

Table 6 presents the 5-fold CV results of all three models on the 13-class datasets. As mentioned above, randomized search CV was used to determine the best hyperparameter values on which to train each model. For SVM, they were $C = 10$ and the *rbf* kernel [69]. For RF, *max_depth* = 70, *max_features* = 'sqrt', *min_samples_leaf* = 1, *min_samples_split* = 10, *n_estimators* = 190, and *bootstrap* = False. Finally, the best hyperparameters for BERT on the 13 classes were *batch_size* = 8, *learning_rate* = 5×10^{-5}, and number of *epochs* = 40. Since our dataset was imbalanced and we had multiple classes, instead of accuracy, we considered f1-score as the determining factor for identifying the best-performing models. In Table 6, we observe that training on Split 2 (see Table 4) produced the best models across all three classifiers. Table 7 presents the classification report of the best-estimating classifiers on the Split 2 dataset.

Table 6. The 5-fold CV accuracies and f1-scores. The best results were obtained on the Split 2 dataset.

Dataset	SVM Baseline		RF Baseline		Fine-Tuned BERT	
	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
Split 0	0.86	0.71	0.81	0.61	0.87	0.77
Split 1	0.84	0.63	0.79	0.58	0.84	0.70
Split 2	**0.85**	**0.72**	**0.84**	**0.74**	**0.92**	**0.87**
Split 3	0.76	0.61	0.73	0.53	0.87	0.81
Split 4	0.79	0.61	0.76	0.56	0.85	0.77
Mean	0.82	0.65	0.78	0.60	0.87	0.78
Standard Deviation (SD)	0.04	0.05	0.04	0.07	0.03	0.06

The SVM classifier for 13 classes had an accuracy of 0.85 and an f1-score of 0.72, with regularization $C = 0$ and *rbf* kernel as the best parameters. The RF classifier performed better with respect to the f1-score of 0.74, but had a slightly smaller accuracy of 0.84. The fine-tuned BERT model, on the other hand, outperformed both baseline classifiers with an accuracy of 0.92 and an f1-score of 0.87. The confusion matrices in Figure 5 of the classifiers

provide more insight into the classification report and the scores that we observe in Table 7.

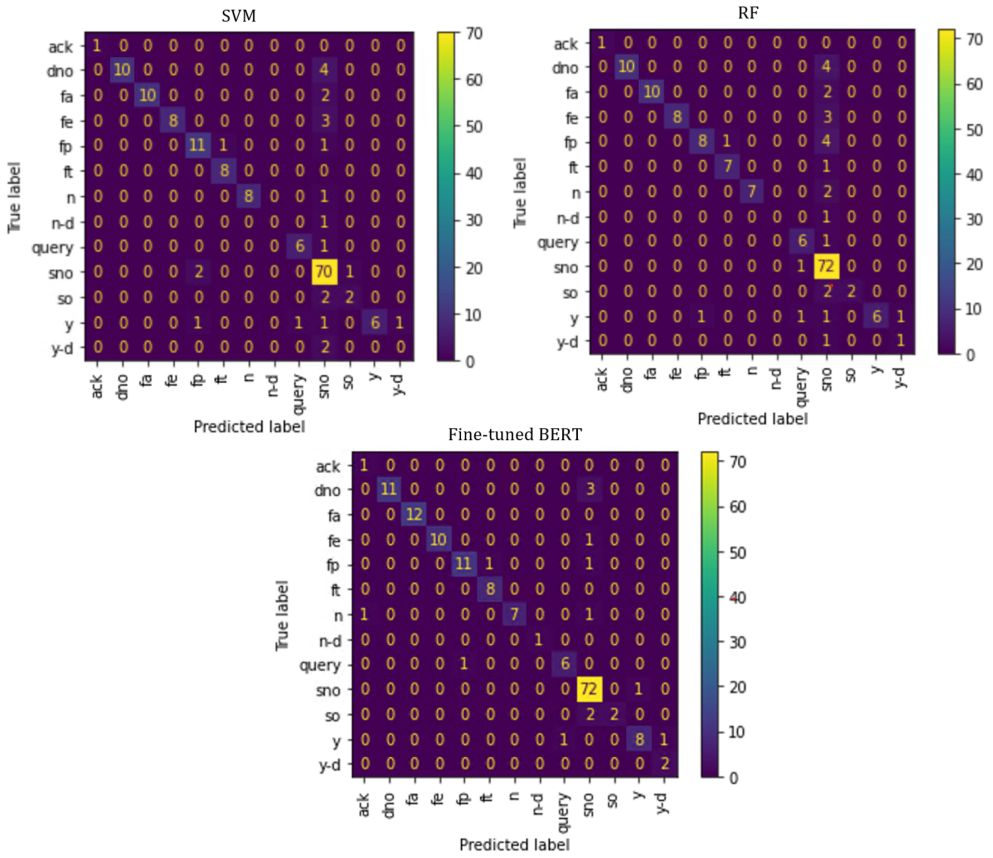

Figure 5. Confusion matrices of SVM (**top left**), RF (**top right**), and fine-tuned BERT (**bottom center**). The color gradient to the right of each matrix corresponds to the number of samples, i.e., the grid cell with $\geq$70 samples will be filled in yellow.

Table 7. Classification report of best classifiers on Split 2.

Dialogue Act	SVM Baseline Accuracy = 0.85 F1-Score = 0.72	RF Baseline Accuracy = 0.84 F1-Score = 0.74	Fine-Tuned BERT Accuracy = 0.92 F1-Score = 0.87
y	0.75	0.75	0.84
y-d	0.00	0.50	0.80
n	0.94	0.88	0.88
n-d	0.00	0.00	1.00
sno	0.87	0.86	0.94
so	0.57	0.67	0.67
ack	1.00	1.00	0.67
dno	0.83	0.83	0.88
query	0.86	0.80	0.86
ft	0.94	0.88	0.94
fa	0.91	0.91	1.00
fe	0.84	0.84	0.95
fp	0.81	0.73	0.88

5. Discussion

From the results in Table 7, we observe that the SVM baseline classifier had an acceptable overall accuracy and f1-score; however, it failed to classify the classes *n-d* (no, with explanation) and *y-d* (yes, with explanation) and performed poorly on *so* (opinionated state-

ments) utterances. On observing the data, an utterance in the *n-d* class looks like "No but I can try", and an utterance in the *n* class looks like "No ma'am". The features for the SVM model did not take into account the order of words, since one-hot encoding was performed by the position of the word in the vocabulary, rather than by the order of occurrence of the words in the utterance (see Appendix A.1 for example). The presence of "No" in both utterances may be why the utterances in *n-d* were misclassified as the *n* label. However, on viewing the confusion matrix for SVM in Figure 5 , we see that the *n-d* sample got misclassified as *sno*. This misclassification was also observed with the *y-d* utterances being misclassified as *sno* class utterances, possibly due to the presence of words common to both classes and/or model bias towards the majority *sno* class; however, there are not enough samples in the test set of the minority classes for an analysis. The RF model performs better for *y-d*, where 1 sample is classified as *sno* (see Figure 5) out of the 2; however, *n-d* is misclassified completely. The performance on *so* improved in the RF classification report. The BERT model's features, on the other hand, involve integer-encoded utterances, where the order of the words in the utterance is preserved and each feature vector is padded to a maximum fixed length (see Appendix A.2, for example). Hence, it performed well on the two classes that the SVM model missed, but the accuracy on *so* remained the same.

Figure 6 shows two graphs where we plot the average accuracy and average f1-score of the three models (see Table 6) with standard error bars to present a visual representation of the variation in results from the 5-fold CV across the three classifiers. Independent one-tailed *t*-tests were conducted between the results of SVM and RF, SVM and fine-tuned BERT, and RF and fine-tuned BERT. The *p*-values are visualized on the dotted lines between the pairs. Results show that there was no statistical significance between the results of SVM and RF (t (4) = 1.25, p = 0.123 for mean accuracy and t (4) = 1.19, p = 0.135 for mean f1-score). However, the results between SVM and fine-tuned BERT (t (4) = −2.11, p = 0.034) for mean accuracy and t (4) = −3.46, p = 0.004 for f1-scores) and RF and fine-tuned BERT (t (4) = −3.56, p = 0.004 for mean accuracy and t (4) = −3.93, p = 0.002 for f1-scores) were statistically significant. The fine-tuned BERT performed significantly better than SVM and RF.

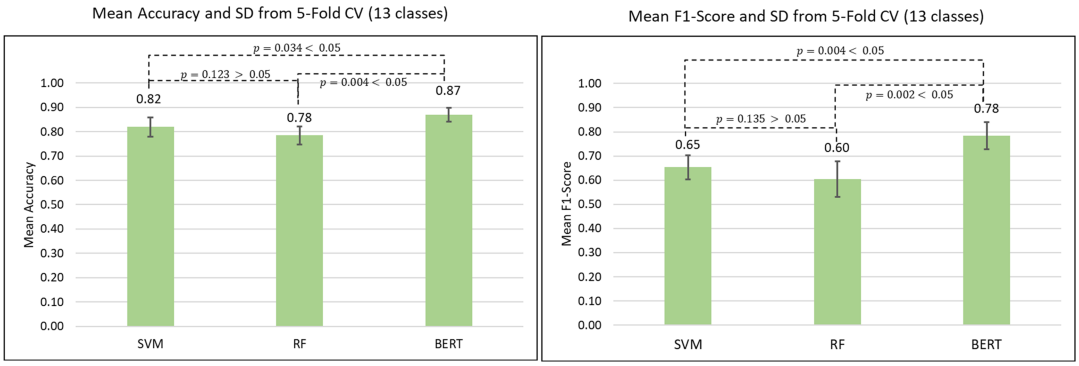

Figure 6. Visualizing accuracies (**left**) and f1-scores (**right**) across all models obtained in 5-fold CV with 13-class data. The *p*-values from the *t*-tests are labeled between SVM and RF, SVM and BERT, and RF and BERT.

In Section 1, we described how our project was motivated by Chakravarty et al. [41], who performed transfer learning and trained three classifiers, including a pretrained BERT, for the task of question answering in an interview context where the input was the question–answer pair. Their DA scheme was adapted from the early work by Jurafsky et al. [81], and led to the development of a rich dataset of interview questions and answers that they used to fine-tune a pretrained BERT with an f1-score of 0.84. Our DA scheme builds on their work and that of Jurafsky et al. [32,40], and achieves similar results, with an f1-score of 0.87. Our score is likely slightly higher due to the difference in the number of classes, and because we were performing single-sentence text classification, which has no dependence

on the question. Their fine-tuned BERT model may have a high f1-score when compared to the scores of the other two models they trained; however, it failed to classify two labels, including one that is similar to *query*. Coincidently, their classifier produced the same f1-score for the *so* label (0.67). Recent research by Wu et al. [58], introduces two further pretrained BERT models for task-oriented dialogue (TOD-BERT) using the BERT-base-uncased model. After pretraining, they fine-tuned their TOD-BERT model for downstream tasks such as DA classification. Although our model had not been further pretrained on other datasets, our micro-f1 score of 0.92 or 92.2% is in line with their DA classification results (91.7%, 93.8%, and 99.5%) achieved on three task-oriented datasets with several domains. These comparisons to previous research further demonstrate the potential of our model to perform well on job interview-related utterances. The results also suggest that further pretraining of our model data will likely improve performance. As for our baseline models, they perform significantly better than those mentioned in [47], which also aimed to classify DA (in the context of online chat forums) using traditional ML methods with 10-fold CV.

6. Conclusions

Autistic individuals face disproportionately poor employment outcomes. Though previous job interview training systems have shown promise, they lack an automated response understanding/labeling mechanism that can be used by the system to automatically classify the types of responses received for a question, regardless of the domain. In this article, we discuss the contributions of our project. First, we present a DA classification scheme in the context of job interviews and provide original data comprising interviewee utterances collected via a virtual job interview training environment. We also share the interview script for use in future data collection studies, which we hope will pave the way for more available data for virtual reality-based job interview training systems for all individuals, including autistic individuals. Second, we present a classifier based on the pretrained BERT, fine-tuned via transfer learning on our original data, that performs with acceptable accuracy across each class, ready to be integrated into a job interview platform for automatic classification of interviewee responses. Automated classification of interviewee utterances in existing job interview training environments can help create adaptive environments where the virtual interviewer (the conversational agent) can understand the basic intent of an interviewee's utterance, which can then be used to create individualized and naturalistic training experiences. The DA detected may also provide insights into the performance of the autistic interviewee to facilitate individualized feedback for improvement. Further, a future developer may experiment with different combinations of classes on which to train their models or further train our fine-tuned BERT, to see what fits best with their application (e.g., another job interview system may only require the utterances from *sno*, *query*, and yes/no type of answers). Appendix A.3 describes the process on how to further train our fine-tuned BERT model for classification.

Despite the above accomplishments, our work has a few limitations. The original dataset, with the addition of more data, is still quite small, and the results reported are on a very small number of samples as support for each class. A larger sample of participants and more training data would be ideal for comparing the efficacy of our ML models. Our ability to carry this out is limited by two key factors: (1) we are sampling a specialized population—working-age, employment-seeking autistic adults; (2) CIRVR currently requires participants to physically come into our lab, where all the multimodal data (responses to questions, eye gaze tracking data, stress detection, etc.) can be collected. We are, however, working to partner with other organizations who work with autistic adults preparing for employment to collect more data by deploying CIRVR at their sites or having these partner organizations help identify a set of participants willing and able to come to our lab. We are also in the early stages of developing a version of CIRVR that can be used on the user's personal device in any location they prefer, which would help in acquiring more data in the future.

As for improvements, to further address the limited training data, we will explore data augmentation methods that have been accepted for use with autistic data [82,83]. It would be useful in future work to examine how a pretrained BERT that has not been fine-tuned on our data performs on the entire dataset. Furthermore, in hopes to improve individual label accuracy of the minority classes, we will explore variations of BERT, such as those in current works [84–86], and explore fine-tuning of TOD-BERT [58] on our data to observe differences in accuracy, and in turn, determine the best-performing method on our data. Future research also needs to evaluate the model's real-world performance by integrating the final BERT classifier trained on the entire dataset, where the predicted labels will be used by the conversational agent to direct the flow of the interview. The interview script and annotation guidelines in Supplementary Materials can aid in replication of our study. The final BERT model has been provided to allow for integration in existing work or for further fine tuning (please contact corresponding author for all data, the model, and the code).

Supplementary Materials: The following are available online at: https://www.mdpi.com/article/10.3390/signals4020019/s1, Data Entry Clerk Script and CIRVR Job Interview Conversation Coding Guidelines.

Author Contributions: Conceptualization, D.A. and N.S.; methodology, D.A. and K.C.; software, D.A.; validation, D.A.; formal analysis, D.A.; investigation, D.A.; resources, N.S. and J.W.; data curation, D.A., K.C., J.W., T.J.V. and M.M.; writing—original draft preparation, D.A.; writing—review and editing, D.A. and N.S.; visualization, D.A.; supervision, N.S.; project administration, J.W., D.A., N.S. and K.C.; funding acquisition, N.S., J.W. and D.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Foundation, grant numbers 1936970 and 2033413. The Article Processing Charge (APC) was funded by the Vanderbilt Award for Doctoral Discovery (VADD).

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki, and approved by the Institutional Review Board (or Ethics Committee) of Vanderbilt University (protocol code #191277 and date of approval 6 August 2019).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study. Written informed consent has been obtained from the patient(s) to publish this paper.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

VR	Virtual Reality
BERT	Bidirectional Encoder Representations from Transformers
DA	Dialogue Act
ASD	Autism Spectrum Disorder
SVM	Support Vector Machine
DAN	Deep Average Network
RNN	Recurrent Neural Network
SwDA	Switchboard Dialogue Act
POS	Parts of Speech
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
RoBERTa	Robustly Optimized BERT Approach
MLM	Masked Language Modeling
NSP	Next Sentence Prediction

TOD-BERT	Task-Oriented Dialogue BERT
CNN	Convolutional Neural Network
CV	Cross-Validation
LSTM	Long Short-Term Memory
CIRVR	Career Interview Readiness in Virtual Reality
IRB	Institutional Review Board
CSV	Comma-Separated Values
NLTK	Natural Language Toolkit
RBF	Radial Basis Function
JSON	JavaScript Object Notation

Appendix A

Appendix A.1

Categorical features such as text in sentences are represented as one-hot encoded vectors. Suppose we have the following utterances spoken by the interviewee or participant:

data = [“I don’t have experience with Microsoft Word”, “I don’t know”]

The vocabulary would then consist of the following words when the above sentences are tokenized and converted to lowercase:

vocab = [“i”, “do”, “n’t”, “have”, “experience”, “with”, “microsoft”, “word”, “know”]

The tokenized data will look like the following:

toknenized_data = [[“i”, “do”, “n’t”, “have”, “experience”, “with”, “microsoft”, “word”], [““i”, “do”, “n’t”, “know”]]

The encoded data will comprise each data sample represented as a list of zeros with the length of the vocabulary. If a token exists in the vocabulary, we mark the presence of the token with a 1 in the position of the word in the vocabulary list (like an “on-and-off” binary switch from 0 to 1). In this example, the vocabulary has nine words; each data sample will start with [0, 0, 0, 0, 0, 0, 0, 0, 0]. The first utterance will be [1, 1, 1, 1, 1, 1, 1, 1, 0], and the second sentence will be represented as [1, 1, 1, 0, 0, 0, 0, 0, 1], where 1 is marked for the presence of the word “know” in the vocabulary. The final encoded data will look like the following:

encoded_data = [[1, 1, 1, 1, 1, 1, 1, 1, 0], [1, 1, 1, 0, 0, 0, 0, 0, 1]]

Since the encoding is carried out by the position of the word in the vocabulary list, the order of words in the utterance is lost.

Appendix A.2

Building on the example from Appendix A.1, integer encoding by the BERT tokenizer maintains the order of words/tokens in an utterance. We define a maximum limit of the length of each feature vector representing each utterance. For this example, let us say that the maximum length of the feature vector is 15 tokens. Since the lengths of the utterances vary, the tokenizer fills the empty space in the feature vector with padding, which represents “no presence of tokens” in those spaces. Each of the words in the vocabulary list above would be assigned a unique integer identifier. Here, we use the position of the word in the vocabulary list to assign the unique identifier for each word, which can be imagined as a hash table. We assign integer 0 for the token <PAD>, which represents padding.

vocab_dictionary = “<PAD>”: 0, “i”: 1, “do”: 2, “n’t”: 3, “have”: 4, “experience”: 5, “with”: 6, “microsoft”: 7, “word”: 8, “know”: 9

Following the same concept, the tokenizer will encode the data according to the dictionary above, where each feature vector is of a maximum length of 15 tokens.

encoded_data = [[1, 2, 3, 4, 5, 6, 7, 8, 0, 0, 0, 0, 0, 0, 0], [1, 2, 3, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

Although we did not need to encode the data ourselves, we have presented the underlying concept of how the BERT model accepts feature vectors.

Appendix A.3. Training a Fine-Tuned BERT

There are several libraries and methods to fine-tune a pretrained BERT model. We use functions from the Hugging Face transformers library to demonstrate the process of training a fine-tuned BERT model. Just as we loaded the *bert-base-uncased* model for fine-tuning, a user can load the fine-tuned model for further pretraining for classification tasks.

The fine-tuned BERT model, when saved, has the following files in a single folder, which we named "fine-tuned-BERT":

- *config.json* has the BERT-base-uncased architecture and configuration of the model, saved in JavaScript Object Notation (JSON).
- *pytorch_model.bin* is the model in a binary file format.
- *special_tokens_map.json* has the [CLS], [SEP], and more special tokens stored as JSON.
- *tokenizer.json* consists of the tokenizer information, as well as the indices of the tokens in the vocabulary.
- *tokenizer_config.json* has the tokenizer configuration of the fine-tuned BERT model.
- *training_args.bin* is a binary file of the training arguments used while training the model.
- *vocab.txt* consists of all the words in the BERT-base vocabulary in a text file.

Below, we describe the process of loading the model for further training/fine-tuning on new data. The programs were written in Python, and the models are stored as PyTorch models that accept PyTorch tensors:

1. Data Preprocessing
   The first step is to prepare the data. Based on the task, a user may choose to preprocess their data, as we describe in Section 3.4. For further fine tuning of BERT, a user may have fewer or more classes than what the model was previously trained to predict. The utterances need to be in text (not tokens) and stored in the first column of a CSV file with the header "text". The labels should be encoded to be represented as unique integers. For example, if your labels are 'positive', 'negative' or 'neutral', you can encode the labels as 'positive': 0, 'negative': 1, 'neutral': 2. The encoded label for each corresponding utterance should be in the second column of the CSV file with the header 'label'. The training and testing data are to be separated into a *training.csv* file and a *testing.csv* file, and the columns in both files should have the same headers "text" and "label" (see Figure A1). Note that the training data can be further split into a training and validation set for evaluation of models during training, which is especially useful for early stopping.

	A	B
1	text	label
2	alright	0
3	ah { personname }	9
4	oh	0
5	yes	11
6	yes	11
7	if you got ta work fast when you	9
8	ok	0
9	coding	9
10	art and science	9

Figure A1. CSV file format for BERT training.

In the program, the CSV files are loaded using the *load_dataset()* function of the Python *datasets* library.
train_data = load_dataset('csv', 'train': 'training.csv')
For tokenization and feature extraction, we use the *BERTFastTokenizer* from the *transformers* library, which has a *from_pretrained()* function that loads the saved tokenizer in fine-tuned BERT. The function takes the model name as an argument or the name

of the folder where the fine-tuned BERT model files are stored, i.e., fine-tuned-BERT. The second parameter is *do_lower_case = True*, which internally converts the text to lowercase if it is not already. Below is the line of code to load the tokenizer.
tokenizer = BERTFastTokenizer.from_pretrained('fine-tuned-BERT', do_lower_case = True)
Once loaded, the dataset is tokenized one sample at a time, where the tokenizer accepts a few arguments: *padding* = [True, False], *truncation* = [True, False], *max_length* = integer value. For example, to make all samples a fixed length, we can set *padding* to True; if the sentences are very long (greater than 512 tokens), we can set *truncation* to True; or if we want each sequence to be of a specific length, e.g., 32 tokens, then we can set *max_length* to that value. The preprocess function passed each sample in the *train_data* and returns the tokenized data.
def *preprocess*(samples): return *tokenizer(samples['text'], padding = True)*
tokenized_train_data = train_data.map(preprocess)
Once the training and testing sets have been tokenized, we move on to the model.

2. Loading the Model
Here, we use the transformers library's *BertForSequenceClassification* to load the model using the function *from_pretrained()*, as shown below.
model = BERTForSequenceClassification.from_pretrained ("fine-tuned-BERT", num_labels = 3)
We can also specify where we want to store the model: on the GPU or on the CPU. To store the model on the GPU, we use *model.to("cuda")*, and to store it on CPU, we use *model.to("cpu")*.

3. Defining Training Arguments
There are several hyperparameters that can be initialized before training. Here, we use the *TrainingArguments* function of the *transformers* library to set the hyperparameters. Specifically, we focus on the number of training epochs or training cycles, the learning rate, and the batch size, which can be affected by the amount of free memory we have available on our system. Here, we had a GPU memory of 16GB, which stores the model and the training data at any given time. Hence, we were only able to initiate the maximum batch size to 32. Other parameters in the *TrainingArgs* can be found at [87]. Just as we conducted hyperparameter tuning using an in-house randomized search on 5 folds of data, the user can follow the same method, or experiment with different values and use early stopping to find the best model. More hyperparameter-tuning options can be found in [88]. The training arguments, for example, are initialized as follows:
training_args = TrainingArguments(
num_train_epochs = 40,
learning_rate = 5e-5,
per_device_batch_size = 8,
weight_decay = 0.01,
output_dir= "Models/fine-tuned-BERT2")

4. Model Training
We use the transformers *Trainer* function to train the model. This function takes a few arguments. The first is the model loaded in Step 2; then, the *args*, which are the *training_args* from Step 3. The *tokenized_train_data* from Step 1 is passed to the *train_dataset* parameter. The *tokenizer* parameter is set to the *tokenizer* that we loaded from the pretrained model. Another optional parameter is the *Data Collator*. Data collators [89] are objects that form batches of data from a list of data input. Here, we used *DataCollatorWithPadding* and passed the *tokenizer* as an argument:
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
trainer = Trainer (model =model,
args = training_args,
train_dataset = tokenized_train_data,
tokenizer = tokenizer,
data_collator = data_collator)

Next, we call the Trainer's train() function.
trainer.train()
After training, the model can be saved along with the learned weights and data using the Trainer's save(model name) function.
trainer.save(model name)

5. Inference
For predictions, we create an inference pipeline. For this task, we can switch to the CPU if not enough GPU space is available. The text input, for example, "Hi there!", is first preprocessed by converting to lower case and by removing any punctuation, as described in Section 3.4. The input is converted into tokens by the tokenizer with the same arguments as those used to tokenize the training set (e.g., padding = True). This input is passed to the model, from which we obtain the outputs on which we apply *SoftMax*, which gives us three probabilities for each class that add up to 1. The argmax of the probabilities is the predicted class label.

References

1. Maenner, M.J.; Shaw, K.A.; Bakian, A.V.; Bilder, D.A.; Durkin, M.S.; Esler, A.; Furnier, S.M.; Hallas, L.; Hall-Lande, J.; Hudson, A.; et al. Prevalence and characteristics of autism spectrum disorder among children aged 8 years—Autism and developmental disabilities monitoring network, 11 sites, United States, 2018. *Mmwr Surveill. Summ.* **2021**, *70*, 1. [CrossRef] [PubMed]
2. American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders: DSM-5*, 5th ed.; American Psychiatric Association: San Francisco, CA, USA, 2013; pp. 50–59. [CrossRef]
3. Taboas, A.; Doepke, K.; Zimmerman, C. Preferences for identity-first versus person-first language in a US sample of autism stakeholders. *Autism* **2023**, *27*, 565–570. [CrossRef] [PubMed]
4. Roux, A.M.; Rast, J.E.; Anderson, K.A.; Shattuck, P.T. *National Autism Indicators Report: Developmental Disability Services and Outcomes in Adulthood*; Life Course Outcomes Program, AJ Drexel Autism Institute, Drexel University: Philadelphia, PA, USA, 2017.
5. Wehman, P.; Taylor, J.; Brooke, V.; Avellone, L.; Whittenburg, H.; Ham, W.; Brooke, A.M.; Carr, S. Toward Competitive Employment for Persons with Intellectual and Developmental Disabilities: What Progress Have We Made and Where Do We Need to Go. *Res. Pract. Pers. Sev. Disabil.* **2018**, *43*, 131–144. [CrossRef]
6. Hayward, S.M.; McVilly, K.R.; Stokes, M.A. Autism and employment: What works. *Res. Autism Spectr. Disord.* **2019**, *60*, 48–58. [CrossRef]
7. Booth, J. *Autism Equality in the Workplace. Removing Barriers and Challenging Discrimination*; Jessica KIngsley Publishers: London, UK, 2016.
8. Harmuth, E.; Silletta, E.; Bailey, A.; Adams, T.; Beck, C.; Barbic, S.P. Barriers and facilitators to employment for adults with autism: A scoping review. *Ann. Int. Occup. Ther.* **2018**, *1*, 31–40. [CrossRef]
9. Ohl, A.; Grice Sheff, M.; Small, S.; Nguyen, J.; Paskor, K.; Zanjirian, A. Predictors of employment status among adults with Autism Spectrum Disorder. *Work* **2017**, *56*, 345–355. [CrossRef]
10. Flower, R.L.; Dickens, L.M.; Hedley, D. Barriers to Employment: Raters' Perceptions of Male Autistic and Non-Autistic Candidates During a Simulated Job Interview and the Impact of Diagnostic Disclosure. *Autism Adulthood* **2021**, *3*, 300–309. [CrossRef]
11. Maras, K.L.; Norris, J.E.; Nicholson, J.; Heasman, B.; Remington, A.; Crane, L. Ameliorating the disadvantage for autistic job seekers: An initial evaluation of adapted employment interview questions. *Autism* **2020**, *25*, 1060–1075. [CrossRef]
12. Smith, M.J.; Ginger, E.J.; Wright, K.; Wright, M.A.; Taylor, J.L.; Humm, L.B.; Olsen, D.E.; Bell, M.D.; Fleming, M.F. Virtual reality job interview training in adults with autism spectrum disorder. *J. Autism Dev. Disord.* **2014**, *44*, 2450–2463. [CrossRef]
13. Haruki, K.; Muraki, Y.; Yamamoto, K.; Lala, D.; Inoue, K.; Kawahara, T. Simultaneous Job Interview System Using Multiple Semi-autonomous Agents. In Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue, Edinburgh, UK, 7–9 September 2022; Association for Computational Linguistics: Edinburgh, UK, 2022; pp. 107–110.
14. Smith, M.J.; Pinto, R.M.; Dawalt, L.; Smith, J.; Sherwood, K.; Miles, R.; Taylor, J.; Hume, K.; Dawkins, T.; Baker-Ericzén, M.; et al. Using community-engaged methods to adapt virtual reality job-interview training for transition-age youth on the autism spectrum. *Res. Autism Spectr. Disord.* **2020**, *71*, 101498. [CrossRef]
15. Baur, T.; Damian, I.; Gebhard, P.; Porayska-Pomsta, K.; André, E. A Job Interview Simulation: Social Cue-Based Interaction with a Virtual Character. In Proceedings of the 2013 International Conference on Social Computing, Alexandria, VA, USA, 8–14 September 2013; pp. 220–227. [CrossRef]
16. Strickland, D.C.; Coles, C.D.; Southern, L.B. JobTIPS: A transition to employment program for individuals with autism spectrum disorders. *J. Autism Dev. Disord.* **2013**, *43*, 2472–2483. [CrossRef]
17. VirtualSpeech: Soft Skills Training with VR, 2021. Available online: https://virtualspeech.com/ (accessed on 12 January 2023).
18. Smith, M.J.; Smith, J.D.; Jordan, N.; Sherwood, K.; McRobert, E.; Ross, B.; Oulvey, E.A.; Atkins, M.S. Virtual Reality Job Interview Training in Transition Services: Results of a Single-Arm, Noncontrolled Effectiveness-Implementation Hybrid Trial. *J. Spec. Educ. Technol.* **2021**, *36*, 3–17. [CrossRef]

19. Smith, M.; Sherwood, K.; Ross, B.; Smith, J.; DaWalt, L.; Bishop, L.; Humm, L.; Elkins, J.; Steacy, C. Virtual interview training for autistic transition age youth: A randomized controlled feasibility and effectiveness trial. *Autism* **2021**, *25*, 1536–1552. [CrossRef]
20. Damian, I.; Baur, T.; Lugrin, B.; Gebhard, P.; Mehlmann, G.; André, E. Games are Better than Books: In-Situ Comparison of an Interactive Job Interview Game with Conventional Training. In *Artificial Intelligence in Education*; Conati, C., Heffernan, N., Mitrovic, A., Verdejo, M.F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 84–94.
21. Schmid Mast, M.; Kleinlogel, E.P.; Tur, B.; Bachmann, M. The future of interpersonal skills development: Immersive virtual reality training with virtual humans. *Hum. Resour. Dev. Q.* **2018**, *29*, 125–141. [CrossRef]
22. Kwon, J.H.; Powell, J.; Chalmers, A. How level of realism influences anxiety in virtual reality environments for a job interview. *Int. J. Hum. Comput. Stud.* **2013**, *71*, 978–987. [CrossRef]
23. Zhao, W. How Different Virtual Reality Environments Influence Job Interview Anxiety. Bachelor's Thesis, The University of Twente, Enschede, The Netherlands, July 2022. Available online: http://essay.utwente.nl/91801/ (accessed on 4 April 2023).
24. Villani, D.; Repetto, C.; Cipresso, P.; Riva, G. May I Experience More Presence in Doing the Same Thing in Virtual Reality than in Reality? An Answer from a Simulated Job Interview. *Interact. Comput.* **2012**, *24*, 265–272. [CrossRef]
25. Adiani, D.; Breen, M.; Migovich, M.; Wade, J.; Hunt, S.; Tauseef, M.; Khan, N.; Colopietro, K.; Lanthier, M.; Swanson, A.; et al. Multimodal job interview simulator for training of autistic individuals. *Assist. Technol.* **2023**, 1–18. [CrossRef]
26. McTear, M.F.; Callejas, Z.; Griol, D. *The Conversational Interface*; Springer: Cham, Switzerland, 2016; Volume 6.
27. Searle, J.R. What is a speech act. *Perspect. Philos. Lang. Concise Anthol.* **1965**, *2000*, 253–268.
28. Chatterjee, A.; Sengupta, S. Intent Mining from past conversations for Conversational Agent. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 8–13 December 2020; International Committee on Computational Linguistics: Barcelona, Spain, 2020; pp. 4140–4152. [CrossRef]
29. Enayet, A.; Sukthankar, G. Poster: A Transfer Learning Approach for Dialogue Act Classification of GitHub Issue Comments. In Proceedings of the International Conference on Social Informatics, Pisa, Italy, 6–9 October 2020. [CrossRef]
30. Montenegro, C.; López Zorrilla, A.; Mikel Olaso, J.; Santana, R.; Justo, R.; Lozano, J.A.; Torres, M.I. A Dialogue-Act Taxonomy for a Virtual Coach Designed to Improve the Life of Elderly. *Multimodal Technol. Interact.* **2019**, *3*, 52. [CrossRef]
31. Ahmadvand, A.; Choi, J.I.; Agichtein, E. Contextual Dialogue Act Classification for Open-Domain Conversational Agents. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 1273–1276. [CrossRef]
32. Stolcke, A.; Ries, K.; Coccaro, N.; Shriberg, E.; Bates, R.A.; Jurafsky, D.; Taylor, P.; Martin, R.; Ess-Dykema, C.V.; Meteer, M. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Comput. Linguist.* **2000**, *26*, 339–373. [CrossRef]
33. Wood, A.; Eberhart, Z.; McMillan, C. Dialogue Act Classification for Virtual Agents for Software Engineers during Debugging. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, New York, NY, USA, 27 June–19 July 2020; pp. 462–469. [CrossRef]
34. Liu, L.; Tang, L.; Dong, W.; Yao, S.; Zhou, W. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus* **2016**, *5*, 1–22. [CrossRef] [PubMed]
35. Khatri, C.; Goel, R.; Hedayatnia, B.; Metanillou, A.; Venkatesh, A.; Gabriel, R.; Mandal, A. Contextual Topic Modeling for Dialog Systems. In Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 18–21 December 2018; pp. 892–899. [CrossRef]
36. Boyer, K.E.; Grafsgaard, J.F.; Ha, E.Y.; Phillips, R.; Lester, J.C. An Affect-Enriched Dialogue Act Classification Model for Task-Oriented Dialogue. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, OR, USA, 19–24 June 2011; pp. 1190–1199.
37. Saha, T.; Gupta, D.; Saha, S.; Bhattacharyya, P. Emotion aided dialogue act classification for task-independent conversations in a multi-modal framework. *Cogn. Comput.* **2021**, *13*, 277–289. [CrossRef]
38. Rangarajan, V.; Bangalore, S.; Narayanan, S. Exploiting prosodic features for dialog act tagging in a discriminative modeling framework. In Proceedings of the Interspeech, Antwerp, Belgium, 27–31 August 2007.
39. Adiani, D.; Itzkovitz, A.; Bian, D.; Katz, H.; Breen, M.; Hunt, S.; Swanson, A.; Vogus, T.J.; Wade, J.; Sarkar, N. Career Interview Readiness in Virtual Reality (CIRVR): A Platform for Simulated Interview Training for Autistic Individuals and Their Employers. *ACM Trans. Access. Comput.* **2022**, *15*, 1–28. [CrossRef]
40. Jurafsky, D.; Shriberg, E.; Biasca, D Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13. University of Colorado at Boulder &+ SRI International, 1997. Available online: https://www1.icsi.berkeley.edu/pubs/speech/tr-97-02.pdf (accessed on 4 April 2023).
41. Chakravarty, S.; Chava, R.V.S.P.; Fox, E.A. Dialog Acts Classification for Question-Answer Corpora. In Proceedings of the Third Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL@ICAIL), Montreal, QC, Canada, 17–21 June 2019. Available online: https://ceur-ws.org/Vol-2385/paper6.pdf (accessed on 4 April 2023).
42. Sadohara, K.; Kojima, H.; Narita, T.; Nihei, M.; Kamata, M.; Onaka, S.; Fujita, Y.; Inoue, T. Sub-lexical Dialogue Act Classification in a Spoken Dialogue System Support for the Elderly with Cognitive Disabilities. In Proceedings of the Fourth Workshop on Speech and Language Processing for Assistive Technologies, Grenoble, France, 21–22 August 2013; pp. 93–98.
43. Fernandez, R.; Picard, R.W. Dialog act classification from prosodic features using support vector machines. In Proceedings of the Speech Prosody 2002, Aix-en-Provence, France, 11–13 April 2002; pp. 291–294.

44. Surendran, D.; Levow, G.A. Dialog act tagging with support vector machines and hidden Markov models. In Proceedings of the 2006 Interspeech, Pittsburgh, PA, USA, 17–21 September 2006.
45. Grau, S.; Sanchis, E.; Castro, M.J.; Vilar, D. Dialogue act classification using a Bayesian approach. In Proceedings of the 9th Conference Speech and Computer, St. Petersburg, Russia, 20–22 September 2004.
46. Keizer, S.; Akker, R.O.D. Dialogue act recognition under uncertainty using Bayesian networks. *Nat. Lang. Eng.* **2007**, *13*, 287–316. [CrossRef]
47. Moldovan, C.; Rus, V.; Graesser, A.C. Automated Speech Act Classification For Online Chat. In Proceedings of the Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, OH, USA, 16–17 April 2011.
48. Fiel, M. Machine learning techniques in dialogue act recognition. *Eest. Raken. Uhingu Aastaraam.* **2007**, *3*, 117–134. [CrossRef]
49. Raheja, V.; Tetreault, J. Dialogue Act Classification with Context-Aware Self-Attention. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; pp. 3727–3733. [CrossRef]
50. The Switchboard Dialog Act Corpus. Available online: https://compprag.christopherpotts.net/swda.html (accessed on 4 April 2023).
51. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; pp. 226–231.
52. Bozinovski, S. Reminder of the First Paper on Transfer Learning in Neural Networks, 1976. *Informatica* **2020**, *44* 291–302. [CrossRef]
53. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for 586 Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (NAACL-HLT 2019), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186. [CrossRef]
54. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
55. Duran, N.; Battle, S.; Smith, J. Sentence encoding for Dialogue Act classification. *Nat. Lang. Eng.* **2021**, 1–30. [CrossRef]
56. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
57. Noble, B.; Maraev, V. Large-scale text pre-training helps with dialogue act recognition, but not without fine-tuning. In Proceedings of the 14th International Conference on Computational Semantics (IWCS), Groningen, The Netherlands, 14–18 June 2021; pp. 166–172.
58. Wu, C.; Hoi, S.C.H.; Socher, R.; Xiong, C. ToD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogues. *arXiv* **2020**, arXiv:2004.06871.
59. Microsoft Azure | Speech-to-Text Documentation. Available online: https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/index-speech-to-text (accessed on 13 January 2023).
60. Xu, B.; Tao, C.; Feng, Z.; Raqui, Y.; Ranwez, S. A Benchmarking on Cloud based Speech-To-Text Services for French Speech and Background Noise Effect. In Proceedings of the 6th National Conference on Practical Applications of Artificial Intelligence, Bordeaux, France, 2021. [CrossRef]
61. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
62. Sun, C.; Qiu, X.; Xu, Y.; Huang, X. How to fine-tune bert for text classification? In Proceedings of the Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, 18–20 October 2019; pp. 194–206.
63. Kim, S.N.; Cavedon, L.; Baldwin, T. Classifying dialogue acts in one-on-one live chats. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Cambridge, MA, USA, 9–11 October 2010; pp. 862–871.
64. Webb, G. Overfitting. In *Encyclopedia of Machine Learning and Data Mining*; Sammut, C.; Webb, G.I., Eds.; Springer: Boston, MA, USA, 2017; pp. 947–948. [CrossRef]
65. Branco, P.; Torgo, L.; Ribeiro, R.P. A Survey of Predictive Modeling on Imbalanced Domains. *ACM Comput. Surv.* **2016**, *49*. [CrossRef]
66. Li, D.; Hasanaj, E.; Li, S. 3-Baselines, 2010. Available online: https://blog.ml.cmu.edu/2020/08/31/3-baselines/ (accessed on 13 January 2023).
67. Suthaharan, S. Support vector machine. In *Machine Learning Models and Algorithms for Big Data Classification*; Springer: Cham, Switzerland, 2016; pp. 207–235. [CrossRef]
68. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
69. Schabus, D.; Krenn, B.; Neubarth, F. Data-Driven Identification of Dialogue Acts in Chat Messages. In Proceedings of the Conference on Natural Language Processing, Bochum, Germany, 19–21 September 2016.
70. Malik, U.; Barange, M.; Saunier, J.; Pauchet, A. Performance comparison of machine learning models trained on manual vs ASR transcriptions for dialogue act annotation. In Proceedings of the 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), Volos, Greece, 5–7 November 2018; pp. 1013–1017.

71. Dantas, J. The Importance of k-Fold Cross-Validation for Model Prediction in Machine Learning. Towards Data Science. 2020. Available online: https://towardsdatascience.com/the-importance-of-k-fold-cross-validation-for-model-prediction-in-machine-learning-4709d3fed2ef (accessed on 4 April 2023).
72. KFold. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html (accessed on 4 April 2023).
73. Bird, S.; Klein, E.; Loper, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2009.
74. Label Encoder. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (accessed on 4 April 2023).
75. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830. [CrossRef]
76. Randomized Search Cross Validation. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html (accessed on 4 April 2023).
77. Sreenivasa, S. Radial Basis Function (RBF) Kernel: The Go-To Kernel. 2020. Available online: https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a (accessed on 4 April 2023).
78. Hugging Face Transformers. Available online: https://huggingface.co/docs/transformers/index (accessed on 4 April 2023).
79. Fast Tokenizer. Available online: https://huggingface.co/learn/nlp-course/chapter6/3 (accessed on 4 April 2023).
80. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
81. Jurafsky, D.; Shriberg, E.; Fox, B.; Curl, T. Lexical, prosodic, and syntactic cues for dialog acts. In *Discourse Relations and Discourse Markers*; Association for Computational Linguistics (ACL): Montreal, QC, Canada, 1998.
82. Shushma, G.; Jacob, I.J. A Semantic Approach for Computing Speech Emotion Text Classification Using Machine Learning Algorithms. In Proceedings of the 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichy, India, 16–18 February 2022; pp. 1–5.
83. Hendr, A.; Ozgunalp, U.; Erbilek Kaya, M. Diagnosis of Autism Spectrum Disorder Using Convolutional Neural Networks. *Electronics* **2023**, *12*, 612. [CrossRef]
84. Wu, T.W.; Su, R.; Juang, B.H. A Context-Aware Hierarchical BERT Fusion Network for Multi-turn Dialog Act Detection. In Proceedings of the 2021 Interspeech, Brno, Czechia, 30 August–3 September 2021.
85. Wu, T.W.; Juang, B.H. Knowledge Augmented Bert Mutual Network in Multi-Turn Spoken Dialogues. In Proceedings of the ICASSP 2022—2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022; pp. 7487–7491.
86. Peng, W.; Hu, Y.; Xing, L.; Xie, Y.; Zhang, X.; Sun, Y. Modeling intention, emotion and external world in dialogue systems. In Proceedings of the ICASSP 2022—2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022; pp. 7042–7046.
87. Hugging Face Training Arguments. Available online: https://huggingface.co/docs/transformers/v4.28.1/en/main_classes/trainer#transformers.TrainingArguments (accessed on 4 April 2023).
88. BERT HyperParameter Tuning. Available online: https://huggingface.co/docs/transformers/hpo_train (accessed on 4 April 2023).
89. Data Collator. Available online: https://huggingface.co/docs/transformers/main_classes/data_collator3 (accessed on 4 April 2023).