Neural Lie Detection with the CSC Deceptive Speech Dataset

Shloka Desai shloka@stanford.edu
Maxwell Siegelman maxsieg@stanford.edu
Zachary Maurer zmaurer@stanford.edu

Abstract

In this project, we explore whether recurrent neural networks (RNNs) can be used to classify a speech segment as truthful or deceptive using the Columbia University-SRI-Colorado State University (CSC) Deceptive Speech Dataset[10]. As a comparative baseline, we use traditional machine learning models like Logistic Regression and SVMs that are representative of previous research on this topic and compare their performance to a variety of different neural network architectures.

1 Introduction and Dataset

The CSC Deceptive Speech corpus is a collection of close-mic'd audio recordings of a conversation between an interviewer and the subject, where the subject is incentivized to mix lies with truths convincingly. Our training examples consist of short speech recordings (2-10 seconds) of the subject answering questions posed by the interviewer. The labels (i.e. truth and lie) were recorded via a hidden pedal that the subject could operate to annotate their answers. In total we spliced approximately transcribed audio for 32 speakers with roughly 100-200 recordings each, which resulted in 4,100 training examples total.

Dataset Statistics	
# Speakers	32
# Training Examples	4,100
% TRUTH	61.39%
% LIE	38.61%

Figure 1: Dataset statistics

The data from the CSC corpus had to be cleaned significantly for our purposes, since the data was provided as a collection of unedited FLAC files and various Praat TextGrid files. First, the TextGrid files which contained pedal-annotation time intervals had to be matched with the intervals in the logically-subdivided, word-level transcriptions. This could be error prone at times because the transcript timings did not always agree with the pedal timings.

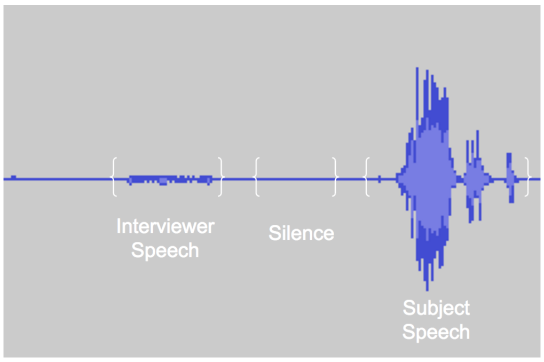


Figure 2: An example of a waveform

After creating the alignments, we spliced the audio files into smaller segments that contained the relevant speech utterances for that training example. After qualitatively reviewing the recording waveforms, these snippets were processed further to remove (1) extraneous silence or (2) background noise (e.g. interviewer speech) and thus isolate (3) the subjects speech. These three components could be identified clearly on a waveform by the difference in loudness levels, and since loudness could vary significantly between subjects, we found the relevant threshold for each audio segment by clustering samples with K-Means with 3 clusters to determine the threshold that would indicate that subject is talking. Silence and interviewer noise coming before the first and after the last occurrences of subject speech were removed.

2 Background and Related Work

The standard for lie detection since the early 20th century has been the polygraph, which records physiological markers like heart rate, blood pressure, breathing rate and perspiration. The tester starts by asking simple questions to assess a baseline for the subject that indicates truthfulness, then the examiner moves on to the actual assessment, during which the subject might lie. This baseline assessment of the subjects physiological reactions is then used to make a judgment about whether the subject is lying or not. The accuracy of the polygraph has been debated. Frank Horvath of the American Polygraph Association says that "Proponents will say the test is about 90 percent accurate. Critics will say its about 70 percent accurate... The process in which the questions are asked and the sequence of the questions may affect how a person reacts"[1].

Speech based analysis has also been attempted since the 1970s, but a number of studies have investigated audio recording based lie detection algorithms and found their efficacy questionable. A 2013 paper by Horvath et al. investigated the effectiveness of a technique called Layered Voice Analysis, a proprietary algorithm based on voice frequency analysis which is marketed by an Israeli company as a tool for fraud detection. The study found LVA to be only 48% accurate on their 72 test subjects[2]. It is interesting to note that both in the case of the polygraph and the voice based analysis there is a evidence from studies like O'Sullivan et al. assert that lie detection capability increases in high-stakes situations, where the subject is under pressure[3]. The Horvath study tried to replicate this by using recordings of actual police polygraph interviews, but in general this could be a confounding factor for general-purpose lie detection. Certainly, the CSC dataset used in this paper was not collected from subjects in high stakes situations, but a financial incentive and self-worth goal was used to incentivize convincing lies.

More specific to our dataset, a handful of studies have used traditional machine learning techniques on the CSC Deceptive Speech Dataset. Hirschberg et al. [4] use a combination of lexical (number of pauses, presence of positive emotion words), prosodic (pitch, energy, speaking rate) and speaker dependent features (subject id, subject gender, and subject-dependent ratios such as ratios of laughter-to-dialogue, cue-phrase frequency and filled pause frequency). They achieve an error rate of 33.6% in identifying examples of deceptive speech using a combination of lexical and acoustic features and a high-dimensional speaker-dependent feature set. However, despite these improvements, their results were extremely close to their baseline. Furthermore, it was unclear from their error analysis whether this was better than chance and what features were most important for their model. Benus et. al [5] explores using filled pauses (ums and uhs) in addition to the features described above. This reduces the error rate to 32.2% – which was criticized as a statistically insignificant improvement by Salvetti [6]. Furthermore, both papers focus on accuracy (error rate) as an evaluation metric, which, due to the moderately unbalanced nature of the classes, might be a misleading way of quantifying model performance.

Finally, there are some major differences in methodology that make our work difficult to compare directly to prior work.

First, Hirschberg et al. split the dataset (transcripts and audio) at a different level of granularity than our work. Their work considered 9491 slash-units which are post-processed subdivisions of contiguous sections where the subject annotated their responses as lie or truth. While this is probably a reasonable design choice, it wasnt clear to us how to mimic this division when starting the project, and thus we decided to use the pedal-annotations as our level of granularity for dividing training examples.

Second, Hirschberg et al. notes they "divided the data 90%/10% into training and test sets five times (with replacement), trained on the former and tested on the latter, and then averaged the results for the numbers presented below". It is unclear from this statement alone, whether or not the training split described by Hirschberg would have resulted in an explicit train-test leak, due to the replacement sampling strategy. Nonetheless, in either case, the researchers did not use a validation set to develop their models, which could still be considered an implicit form of train-test leak.

Third, we felt that the most relevant extrinsic task to compare our classifier to would be a polygraph. In a polygraph assessment, a baseline can only reliably be established on known, truthful statements. The speaker dependent profiles that

Hirschberg et al. used to gain their noted decrease in error relied on developing a speaker-dependent profile that also included all (truth and lie) examples in their dataset. We believed this design choice to be unrealistic, and also possibly leading to a train-test leak, due to the high-dimensionality of the speaker profiles.

3 Features and Methods

Feature Extraction

Overall, we extracted two types of features: acoustic and lexical.

The acoustic features in the table to the right were extracted using a 50% overlapping, sliding-window sampling approach with frame sizes of 50msecs and 100msecs with the PyAudioAnalysis[7] software package. The neural classifiers were given these features as time-sequences. However, for the linear baselines, we summarized each of examples corresponding time sequence as a series of distributional statistics (mean, median, skew, max, min, etc.). A PCA visualization of these summarized features follows below. Although PCA visualizations are imprecise for visualizing higher-dimensional data, there is nothing in this plot that would suggest a straightforward, natural clustering in the data.

	Feature
1	Zero Crossing Rate
2	Energy
3	Entropy of Energy
4	Spectral Centroid
5	Spectral Spread
6	Spectral Entropy
7	Spectral Flux
8	Spectral Rolloff
9-21	MFCCs
22-33	Chroma vector
34	Chroma deviation

Figure 3: Features extracted

In addition to these basic features, we also developed a speaker-dependent audio feature-set which was meant to replicate the baseline readings taken by a conventional polygraph. In that scenario, a subject is asked to speak about a known subject truthfully (such that the examiner can corroborate veracity) in order to understand

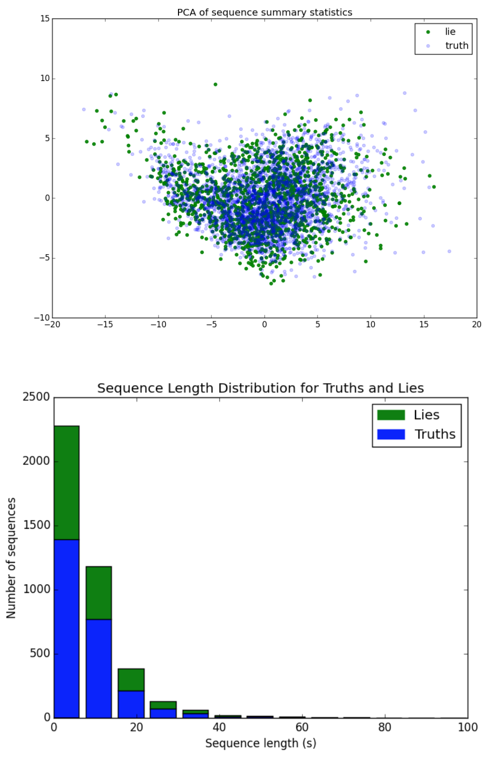


Figure 4: PCA visualization of sequence summary statistics and sequence length distributions for truths and lies

the baseline biometrics that indicate truth-telling on a polygraph machine. We tried to factor this approach into our feature extraction process by computing the mean of all the features mentioned above over all utterances labelled as Truth for each speaker. This yielded 32 (the number of subjects) speaker profiles. Then, we augmented our original acoustic features by appending the relative difference between each feature at each timestep and the mean per speaker for all examples. We will refer to these as speaker-dependent audio features in future sections. Each acoustic feature frame was 34 dimensional and each speaker-dependent frame was 68 dimensional.

Lexical features were encoded using GloVe Wikipedia and CommonCrawl 100-dimensional embeddings[9] based on the transcripts provided with the dataset.

After plotting the cumulative distribution of transcript lengths and acoustic feature vector lengths, we decided truncate (or zero-pad) all transcript lengths to 27 tokens and acoustic features to 150 frames. This corresponds to capturing the entirety of roughly 85-90% of all training examples. We chose to truncate to shorter sequences avoid biasing the model towards outliers and leftover artifacts from the data-cleaning. A plotted distribution of audio-sequence length is above.

3.1 Baseline Classifiers

We experimented with three separate baseline classifiers. The first was an SVM classifiers with an RBF-Kernel. Each SVM varied only in the features that were used as input. Each was separately cross-validated over the slack-parameter gamma, but little difference was observed after tuning this hyperparameter.

We also experimented with using a KNN classifier on the same sets of features as the SVM. We only evaluated each test sample against training samples from the same speaker, meaning that the classification for a test sample was actually the majority vote of training samples from the same speaker. Tuning the hyperparameter k didnt seem to matter for k between 5 and 10, so we used k=5.

(Logistic Regression was also tried, but no performance difference from an SVM was noted.)

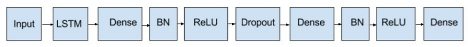
3.2 Neural Classifiers

We used a number of different simple neural networks that used LSTM cells as the primary recurrent neural unit. ReLU nonlinearities were used in between layers and Batch Norm and Dropout were used as regularizers. All neural networks were implemented in PyTorch[8], optimized with an Adam Optimizer (lr = 1e-3) and trained using a Binary Cross Entropy loss.

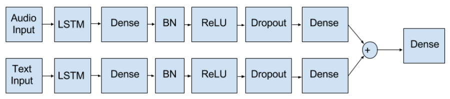
Here are the key differences between the models:

1. Simple Audio RNN: Acoustic features fed as input to an LSTM, only the final hidden state is flattened and fed to a linear classification layer.

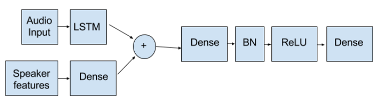
2. Complex Audio RNN: Speaker-dependent acoustic features are used as input to an LSTM. For each example, all hidden states for all timesteps are flattened and then through two decreasing-size Fully Connected-BatchNorm-ReLU blocks (FC-BN-ReLU) to a final classification layer.



3. Hybrid RNN: Speaker-dependent features and lexical features are encoded in separate LSTMs and passed through 2 separate FC-BN-ReLU blocks. The resulting activations are concatenated and passed through a final classification layer.



A variant of the above model was explored by combining the representations earlier in the network as shown below, but this model did not demonstrate any performance differences and was excluded from further considerations.



4. Lexical RNN: Same structure as Hybrid RNN, but only lexical features were used as inputs.

4 Results and Discussion

To evaluate the models without sacrificing too much training data, the following methodology was used: one speaker had all of their data held out, and the remaining data was split into a training set of 3600 samples with a validation set of the remaining data (between 200 and 400 samples). The model was then trained on the training set, and during training the version of the model that performed the best on the validation set was saved. This best version of the model was then evaluated on the test data. Each of the 32 speakers was held out in this way, and the results over all of the models were averaged across all held-out subjects.

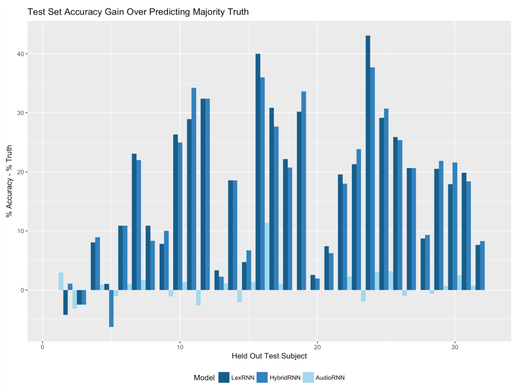
The results obtained from our models via this process shown are below. Our baseline models failed to perform higher than the percentage of the majority class, as they defaulted to predicting ”Truth” in almost all cases. The test set accuracy of the SVM was 62% and the KNN had an accuracy of 60%. The table below show that RNNs relying purely on acoustic features (not speaker dependent) were unable to ever exceed the percentage of Truth in the data by more than about 2%. With different hyperparameters RNNs we were able to overfit on the training data, but no pure audio model ever achieved better performance on the validation and test sets than 62%. This result is comparable to those of Hirschbergs acoustic-only classifiers.

Hybrid RNNs and Lexical RNNs performed very similarly on the test set, despite the lexical RNN slightly outperforming the hybrid RNN on the validation sets and the hybrid RNN outperforming the lexical RNN on the training data. These models outperform the default behavior of guessing the majority class - Truth - on every sample and demonstrate an improvement over previous research benchmarks.

Model	Dataset	% Accuracy	% Truth	% Lie
LexRNN	Test	78.60976	61.85095	38.14905
HybridRNN	Test	78.41463	61.85095	38.14905
AudioRNN	Test	62.21951	61.85095	38.14905

Model	Dataset	% Accuracy	% Truth	% Lie
LexRNN	Validation	75.89076	61.54339	38.45661
HybridRNN	Validation	73.99160	61.67919	38.32081
AudioRNN	Validation	61.24370	61.41194	38.58806

Figure 5: Test and Validation Results



Model	Dataset	% Accuracy	% Truth	% Lie
HybridRNN	Training	84.11372	61.36719	38.63281
LexRNN	Training	81.03385	61.37847	38.62153
AudioRNN	Training	62.59115	61.40278	38.59722

Figure 6: Training Results

The below graph displays the performance of the models compared to guessing all Truth broken down by speaker. It is clear that the AudioRNN does not significantly outperform guessing all Truth on any of the speakers. On the other hand, the LexRNN and HybridRNN both significantly outperform guessing all Truth on a large majority of the speakers. Further, the LexRNN and HybridRNN generally perform well across all held-out test subjects. A significant difference between the pure LexRNN and the HybridRNN is only present on five or six of the speakers.

The graph also shows it was hard for any of the models to do better than guessing all Truth on some of the speakers . Specifically, on five of the speakers none of the models were able to perform significantly above guessing all Truth. However, this is most likely because the percentage of Truth samples for those speakers is very high. For example, speaker 1 had greater than 90% of their audio samples labeled as Truth. Thus, in some cases, this graph does not indicate that some speakers were able to fool the models more than others, but instead the unbalanced nature of some of the speakers data.

The confusion matrix for all three models confirms why the AudioRNN never achieved an accuracy greater than the percentage of Truth by more than a few percent. The audio only model essentially never labeled samples as a Lie; recall on lies was only 7.7%, while precision was 58.1%. Experiments with weighting the classes to avoid this were unhelpful, as they quickly resulted in over-

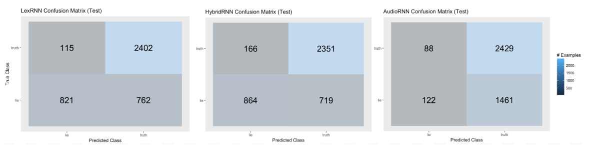


Figure 7: Confusion matrices for the different models

fitting.

Meanwhile, the confusion matrices of the HybridRNN and the LexRNN show significantly improved recall and precision on lies, along with a corresponding improvement in precision on truths. Interestingly, the HybridRNN classifies more samples as lies overall, getting a recall and precision of 54.5% and 83.8%, while the LexRNN is more likely to predict truth in general and thus gets recall and precision of 51.8% and 87.7%. This means that while the lexical and hybrid RNNs had nearly exactly the same accuracy overall, the HybridRNN traded some precision on lies in order to achieve higher recall. Whether these results would be consistent over a larger dataset would be an interesting direction for future work.

In response to these results, we tried to understand whether LexRNN and HybridRNN was actually predicting based on a very simple estimator present in the transcript data. We singled out two main possible features: (1) disfluency frequency and (2) transcript length. After receiving these results, we then trained an SVM on just disfluency frequencies (”uh” and ”um”, standardized across transcripts) and the number of tokens in each transcript. We found that this SVM did not produce any interesting results, and performed at our baseline level of predicting the majority class, in almost all cases. This conclusion was further confirmed by plotting the distribution of lies and truths by sequence length and transcript length; as no noticeable trends seems present.

Thus, it proved difficult to get good results from training an RNN with pure audio data. The AudioRNN only slightly outperformed predicting all Truth, and barely exceeded the baseline SVM performance. This result is not necessarily surprising given the lack of success in previous audio based approaches to lie detection and the nature of our dataset.

That said, it was interesting to note that a RNN based on lexical features could out-perform existing benchmarks on the corpus. Hirschberg et al. did notice in their initial observations about the data, that increases in pleasantness of affect, according to Whissels s Dictionary of Affect in Language[11] was associated with Lie examples. It is possible that the GloVe-based embeddings are capturing some higher-dimensional form of semantic and emotional effect, which may indicate deceptive behavior.

5 Conclusion

Overall, the results are not encouraging for lie detection based solely on acoustic features. Combining acoustic features with lexical features didn’t yield higher accuracy than lexical-feature-only models, and the only potentially interesting result from the hybrid networks is that it may result in a recall-precision tradeoff.

If one compares our classifier to a polygraph, the success of lexical features over audio features could be viewed as counterintuitive. A polygraph is designed to successfully measure involuntary biological responses. Speech patterns, captured by audio features, seem more subconscious and comparable to the inputs of a polygraph, than lexical patterns which often seemed to be assumed to be conscious activity. However, this position is purely a matter of opinion.

Three possible avenues for future research are suggested by our work.

First, extend our method for creating a speaker-dependent profile that more accurately captures a baseline for each subject. We only used a simple average and then appended the relative difference at each timestep, it would be possible to calculate more distributional statistics to provide more information to the classifier.

Second, since the CSC dataset is relatively small, it would obviously be beneficial to collect more data and extend the diversity of the dataset. More data, in particular, would benefit neural network models, which generally require more training examples to converge. Although probably impractical, it would be interesting to explore whether broadcast television interview audio from the internet could annotated with Lie and Truth via crowdsourcing. A shorter-term option, would be to reproduce the subdivision preprocessing done by Hirschberg, which would effectively double the number of examples.

Third, it seems highly relevant to research the benefits of using lexical embeddings like GloVe to predict lies and truths. Using semi-supervised word embeddings seemed to outperform existing hand-crafted lexical features (e.g. cue phrases, disfluencies, etc.) used in previous research. Isolating the relevant lexical features, may also provide future speech research more specific avenues to direct its focus.

References

1. http://abcnews.go.com/US/story?id=92847&page=1
2. Horvath, F., McCloughan, J., Weatherman, D., & Slowik, S. (2013). The Accuracy of auditors' and layered voice Analysis (LVA) operators' judgments of truth and Deception During Police Questioning. Journal of forensic sciences, 58, 385-392
3. O'Sullivan, Maureen; Frank, Mark G.; Hurley, Carolyn M.Tiwana, Jaspreet,(2009). Police lie detection accuracy: The effect of lie scenario. Law and Human Behavior, Vol 33(6), 530-538.
4. Hirschberg, Julia, et al. ”Distinguishing deceptive from non-deceptive speech.” Interspeech. 2005.
5. Benus, Stefan, et al. ”Pauses in deceptive speech.” Speech Prosody. Vol. 18. 2006.
6. Salvetti, Franco, ”Detecting Deception in Text: A Corpus-Driven Approach” (2012). Computer Science Graduate Theses & Dissertations. Paper 42.
7. Theodoros Giannakopoulos, pyAudioAnalysis: https://github.com/tyiannak/pyAudioAnalysis
8. https://github.com/pytorch/pytorch
9. https://nlp.stanford.edu/projects/glove/