Detecting Lies via Speech Patterns

Amanda Chow
Department of Computer Science
Stanford University
amdchow@stanford.edu

John Louie
Department of Computer Science
Stanford University
jwlouie@stanford.edu

Abstract

The objective of this project is to use speech and natural language processing techniques to determine whether an individual is lying, given an audio recording of them speaking. Using the Columbia-SRI-Colorado (CSC) corpus, we extracted accoustic, lexical, and prosodic features and fed them into several machine learning models (e.g. logistic regression, a support vector classifier, and a recurrent neural network) to search for speech patterns correlated with lying. Our analysis, as well as that of researchers before us, shows that the task is extremely difficult given only audio signals, and any models require significant refinement before its performance can warrant practical usage.

1 Introduction

With a growing interest in security, the desire for automated lie detection has long been an endeavor sought after for its applications to court decisions, law enforcement, etc. A widely accepted technique for lie detection include polygraph tests, which rely on the subject's physiological signals/responses to determine the truth or deception in their speech. In addition to advances in lie detection, research in automatic sentiment and emotion detection using speech patterns has gained significant traction in recent years, and has many potential applications, ranging from detecting depression and stress to frustration and anger (Maas, 2017).

2 Background/Related Work

While speech processing methods have advanced steadily over the past years, there is a scarcity of research in detecting deceptive speech that combines speech processing with more advanced machine learning techniques. Studies in deception detection have predominantly been performed from a psychological perspective, often considering multiple indicators such as visual, biometric and physical cues (Pérez-Rosas et al., 2014). In order to apply more advanced speech processing techniques to determine whether it is possible to detect lies solely through audio signals (without biometric signals), we elected to utilize the Columbia-SRI-Colorado (CSC) corpus, which was first developed in 2005 by researchers at Columbia University.

2.1 The CSC Dataset

The CSC corpus consists of 32 hours of speech data from 32 English speakers. Each subject was given the task to convince the interviewer that they possessed the characteristics of a successful entrepreneur. Anytime they answered the interviewer's question with a lie, the subject would press a pedal under the table indicating the truth value of their statement. After the interviews, the interviews were transcribed, and the time ranges for each spoken word and even each phoneme were manually labeled (Hirschberg et al., 2005).

The CSC corpus is particularly interesting because it is one of a few lie detection datasets which does not subject the individuals to what was coined by Hirschberg et al. as "high stakes" scenarios; high stakes situations are those where the subject is very likely to experience fear or shame, such as testifying in court after swearing on the Bible (Hirschberg et al., 2005). Instead, the students were put into a situation where self-promotion was advantageous and financial gain was an incentive. Using this self-promotional interview setup, the researchers created situations in which the subjects were encouraged to lie or tell the truth in order to score well in their interview

(Hirschberg et al., 2005).

2.2 Previous Work

Using this dataset, Hirschberg et al. performed a preliminary pass at extracting primarily lexical (e.g. part-of-speech, tenses, etc.), prosodic and acoustic features and used these features for analysis and classification (using the *Ripper* rule-induction classifier). The researchers' predictive analysis primarily utilized sentence-like units, or SUs, which are segments of text that are denoted with punctuations in the subjects' audio recordings; the data collected by Hirschberg et al. yields 9491 usable SUs. The researchers achieved a baseline error rate of 39.8% and were only able to marginally improve their error rate with further feature construction/selection. By including acoustic/prosodic features, the team was able to lower the error rate to 38.5% and by permuting their chosen features, the team was able to obtain an error rate of 37.2%. The team's optimal error rate achieved was 33.6% by using a combination of lexical, prosodic, acoustic and speaker-dependent features (Hirschberg et al., 2005).

Further research by Graciarena et al., a subset of the researchers associated with the Hirschberg et al. team, expanded upon research with the CSC corpus, utilizing more advanced techniques as well as a richer set of features extracted from the corpus (Graciarena et al., 2006). Some of the target features that Graciarena et al. extracted for their analysis were prosodic features like pitch and energy and lexical features from the recording transcriptions like positive and negative emotion words (Graciarena et al., 2006). These prosodic and lexical features (235 features in total) were then fed into a Support Vector Machine (SVM) classifier. Additionally, Graciarena et al. utilized an acoustic Gaussian Mixture Model (GMM) classifier to determine whether acoustic features (e.g. MFCCs, etc.) were useful in determining truthful or deceptive speech. The researchers also proposed a combined classifier, using both SVM and GMM models, to determine whether they could obtain a more accurate classifier. Using their combined classifier, the researchers were able to obtain their best error rate, 35.6%, which exhibits only a marginal improvement over the baseline results determined by Hirschberg et al.

A recurring theme in papers that utilize the CSC corpus appears to be improvements in feature selection methods. For example, in 2008, Torres et al. elected to extract glottal waveform features from the CSC corpus utilizing a combination of the RAPT pitch estimation, DYPSA and Rank-Based Glottal Quality Assessment algorithms (Torres et al., 2008). Utilizing the extracted glottal waveform features, the researchers then performed speaker-dependent feature selection and classification; however, because of the highly individualized nature of people's glottal features, the results were difficult to extrapolate into globally relevant glottal features (Torres et al., 2008).

Additional papers, including the work of Enos et al., take a more quantitative linguistics approach to utilizing the CSC corpus. In the work of Enos et al., the researchers utilized "critical segments" of the corpus in order to determine the global truthfulness of an individual's statement (Enos et al., 2007). However, because of the nature of how critical segments are defined, the researchers had to analyze the corpus and label/mark critical sections by hand. While such a method is useful and reliable if done correctly, this introduces subjectivity into the data analysis since the researchers must be the ones to determine whether a given passage/segment of audio qualifies as a critical segment or not.

3 Approach

3.1 Data Preprocessing

Following the work of previous research, we began preprocessing by dividing interviewees' recordings into their individual question and answer responses, and labeling each clip. The 32 hours of 32 interviews were divide into 4100 labeled clips. Rather than using both the interviewer and the subject's audio, we decided to use only the subject's audio (the right audio channel) because our primary interest was to find patterns in the speech signals of a speaker who may be lying. After extracting the right audio channel, the clips then had long silences where the interviewer was speaking, and some of the clips were over a minute long.

The silences at the start and end of each clip were unnecessary because the subject was not speaking, so we clipped the surrounding silences using the transcript (from the start of the first word to the end of the last word in each clip). After clipping the surrounding silences, the distribution

of clip lengths was more reasonable, with 97% of clips shorter than 20 seconds.

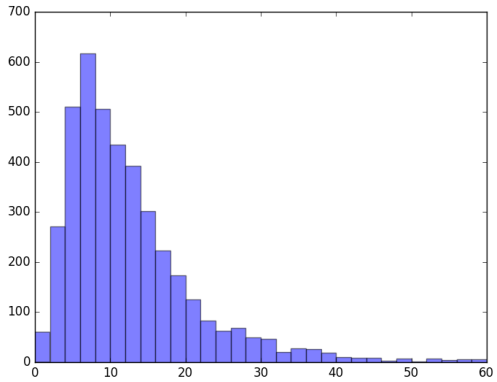

Figure 1: Distribution of clip durations

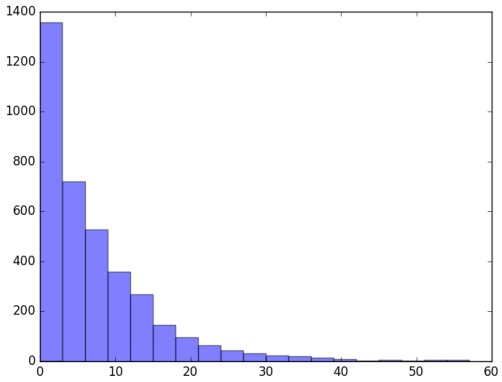

Figure 2: Distribution of clip durations after stripping start and end silence

3.2 Feature Extraction

To start off our baseline model, we extracted the MFCC features from the silence-stripped clips using the `python_speech_features` Python library. However, because each clip had a different duration, and therefore a variable number of samples, we needed some method to normalizing for the varying lengths that the MFCC outputs. There are multiple strategies one can use to normalize the lengths, such as padding, compressing, smoothing, etc.; for the purposes of our baseline, we elected to use a bucketing approach where we would split each clip into 100 timesteps. For example, if we have a recording of 20,000 samples, we would calculate the first MFCC feature values of the first 200 samples and label these values as the first bucket, and so on. With this data, we were then able to run the data through basic learning models for classification. After preprocessing, we had 4100 examples to use for training/validation. Our baseline results are detailed below in Experiments.

For our subsequent models, we extracted different combinations of more advanced features, detailed below:

Acoustic & Prosodic Features

For our subsequent attempts, we opted to use the openSMILE software suite for feature extraction. openSMILE, developed in Munich, Germany, is a library used primarily for feature extraction for large scale audio analysis and is capable of extracting features like signal strength, loudness, MFCC features, pitch, PLP-CC features, and more (Eyben et al., 2013). The library contains different feature sets tailored to different purposes. For the purpose of this task, we opted to utilize one of the software's preconfigured emotion feature sets, the INTERSPEECH 2009 Emotion Challenge feature set, which includes acoustic and prosodic features such as MFCCs, energy, F_0 (the fundamental frequency computed from the Cepstrum), and various contour features.

Lexical Features

We also extracted basic lexical features using the provided interview transcripts for 30 of the 32 subjects. Although the goal was to use only speech input, we considered lexical features to fall in the scope of this task, because there are very sophisticated speech-to-text transcribers that work well in real-time. We essentially mapped the transcripts to the clips we initially parsed, and formulated groupings of question and answer pairings. Once we had individual interviewee answers with associated truth values, we averaged the GloVe word vectors (Pennington et al., 2014) for all the words spoke in each clip to obtain the lexical feature set for any clip. We experimented with different sized GloVe dimensions (50, 100, 200 and 300).

4 Models

4.1 Basic models

Once we combined the acoustic, prosodic and lexical features into a single data set, we ran our data through various learning models. We again uti-

lized `scikit-learn`'s implementations of logistic regression and support vector classifier, but also included gradient boosting classifier and bagging classifier. Additionally, we utilized two neural network implementations: `scikit-learn`'s multi-layer perceptron classifier and a recurrent neural network implemented in Tensorflow. We elected to primarily rely on the performance of simpler models (e.g. logistic regression, SVC, etc.), because the amount of raw data that we were working with (a few thousand training examples), is not conducive to strong or reliable neural network performance and very prone to overfitting.

For each of the classifiers, we utilized feature selection methods, e.g. variance thresholding where we eliminate features that have variance below a specified threshold, as well as regularization (implementation dependent) in an attempt to avoid model overfitting. Lastly, because of the skew of the data, roughly 60% of the learning examples were "true" statements, we utilized dataset balancing methods like SMOTE and random over sampling in an attempt to improve our models' ability to learn.

4.2 Recurrent Neural Network

The second architecture we used was a recurrent neural network (RNN). RNN's are very suitable for speech data, because they detect temporal patterns in data even if the data does not have the same number of timesteps (a problem we mentioned above when extracting MFCC features, because each clip had a different duration). We divided each clip into timesteps of 0.2s, and clipped the longer samples at 20s (only 3% of the dataset was affected by the clipping). Thus, the number of timesteps for each input was capped at a maximum of 100 timesteps. Because 100 is a very large number of timesteps, we used a special type of RNN cell called the Long Short Term Memory cell (Hochreiter and Schmidhuber, 1997), which is capable of remembering long-term patterns better than basic RNN cells (which lose memory and are more prone to vanishing or exploding gradients). Since our dataset is small and neural networks are very prone to overfitting, we also implemented dropout (Srivastava et al., 2014) with a probability of remaining of 0.9. Thus, the final model was a single-level, single-directional LSTM with dropout. At each timestep, we extracted the openSMILE (MFCC and prosodic features) of the 0.2s clip, for 384 features at each timestep. We did not use lexical features at each timestep, because they do not correspond well to the evenly divided timesteps.

4.3 Evaluation of Models

To evaluate the performance of our models, we used was classification accuracy (whether the class that was predicted for an example from the test set was the correct class). Since there was a slightly skewed split of our data between the two classes, we also used the F_1 score (the harmonic mean of precision and recall). For each of the metrics we used k-fold cross validation for more representative/accurate scores. In addition, we ran other evaluations, such as plotting learning curves for the `scikit-learn` learning models, in order to see how varying amounts of training data affect the models' ability to learn (plots and analysis provided below).

5 Experiments

5.1 Baseline Results

With an 80-20 training/test split, the baseline logistic regression model achieved a 54% accuracy. This is slightly better than random, but the model clearly has a lot of room for improvement. When we changed models and used an SVM, we were able to slightly improve our accuracy, obtaining an accuracy of 58%. Note that a naive implementation that guesses true for every clip obtains an accuracy of 61%. These results indicated that we most likely had to reevaluate our data preprocessing methods. We may also have made inaccurate assumptions about the behavior of specific libraries (e.g. our initial MFCC feature extractor) and should investigate their behavior.

5.2 Subsequent Results

In Table 1, we see a subset of the experiments that were run with the acoustic, prosodic and lexical features as well as other techniques like over sampling. The scores in the table are the optimal scores obtained given the various combinations of lexical features and regularization/hyperparameter values tested. Note that additional tests were made with over sampling to account for the skew of the data set and thus the number of examples for the first two sections differs from the number of examples used for training/validation in the latter two sections (see caption for details).

MODEL	ACCURACY	F_1 SCORE
LR (w/o lexical features & regularization)	0.584883 (+/- 0.06)	0.751548
SVC (w/o lexical features & regularization)	0.619433 (+/- 0.00)	0.763593
MLP (w/o lexical features & regularization)	0.619433 (+/- 0.00)	0.763593
GB (w/o lexical features & regularization)	0.502861 (+/- 0.14)	0.739816
BAG (w/o lexical features & regularization)	0.453476 (+/- 0.14)	0.662156
RNN (w/o lexical features & regularization)	0.612685	0.634573
LR (w/ lexical features & regularization)	0.584343 (+/- 0.06)	0.751548
SVC (w/ lexical features & regularization)	0.619433 (+/- 0.00)	0.763593
MLP (w/ lexical features & regularization)	0.619433 (+/- 0.00)	0.763593
GB (w/ lexical features & regularization)	0.506095 (+/- 0.14)	0.744775
BAG (w/ lexical features & regularization)	0.459684 (+/- 0.13)	0.666024
LR (w/ lexical features & regularization; SMOTE over sampling)	0.444700 (+/- 0.08)	0.562962
SVC (w/ lexical features & regularization; SMOTE over sampling)	0.511338 (+/- 0.01)	0.512323
MLP (w/ lexical features & regularization; SMOTE over sampling)	0.496097 (+/- 0.05)	0.529938
GB (w/ lexical features & regularization; SMOTE over sampling)	0.440905 (+/- 0.20)	0.636095
BAG (w/ lexical features & regularization; SMOTE over sampling)	0.504915 (+/- 0.15)	0.584052
LR (w/ lexical features & regularization; random over sampling)	0.428767 (+/- 0.07)	0.550014
SVC (w/ lexical features & regularization; random over sampling)	0.575495 (+/- 0.15)	0.389784
MLP (w/ lexical features & regularization; random over sampling)	0.489122 (+/- 0.05)	0.483429
GB (w/ lexical features & regularization; random over sampling)	0.473527 (+/- 0.13)	0.664981
BAG (w/ lexical features & regularization; random over sampling)	0.584615 (+/- 0.11)	0.662976

Table 1: LR = Logistic regression, SVC = support vector classifier, GB = gradient boosting classifier, MLP = multi-layer perceptron classifier, BAG = bagging classifier and RNN = recurrent neural network; for the first two sections, the number of examples used for training/validation was 3705 and exhibited a 60:40 class split; for the last two sections, the number of examples used for training/validation was 4590 and exhibited a 50:50 class split; accuracy values displayed with 95% confidence interval.

6 Conclusions

6.1 Discussion

Analyzing the results that we obtained from further experimentation with and without lexical features, we see that accuracy performance was generally underwhelming, especially given the fact that the data set initially started with an approximately 60:40 split in class labels. Thus, our model is most likely simply predicting one class for each fold in cross validation. F_1 scores exhibit a slightly more optimistic assessment, yielding a relatively good score (close to 1).

On the other hand, the results from datasets with over sampling paint a slightly more optimistic picture. Our best performance with random sampling and our bagging classifier achieved an accuracy of .58 and an F_1 score of .66. While these kinds of accuracy levels are still far from being usable in a practical sense, if we take a look at the learning curve for the various models, refer to Figures 3 and 4, we see that our bagging and gradient boosting classifiers have the potential to benefit from the inclusion of more data; more data would allow these model to generalized better and to potentially converge to a higher accuracy level.

As for the RNN, it does not perform as well as the linear classifiers, even after hyperparameter tuning. However, the RNN does not seem to be overfitting the data because the training accuracy hovers around 60%. Because we are not overfitting, this suggests that there are not enough training examples for the LSTM to detect patterns in the features - a problem that we predicted earlier when splitting the data into 4000 clips. Other possible sources of error include implementation bugs, inconsistent data, etc.

In general, while we were able to achieve performance comparable to that of previous models run on the dataset, we believe that a significant amount of error in our work can be attributed to the small number of training examples, the split of the two classes (and thus the three classes, splitting lies into the lying up and down classes), and

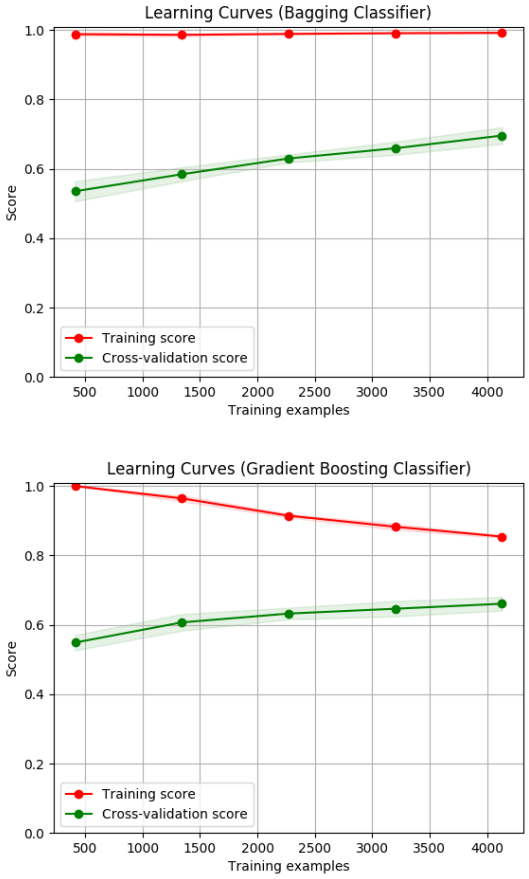

Figure 3: Learning curves for both our bagging classifier (the above figure) and our gradient boosting classifier. Both learning curves were generated using acoustic, prosodic and lexical features, with variance thresholding as well as random over sampling to have a 50:50 class split.

the interview/recording clip delineation methodology. Because the number of training examples was under 4000, the kinds of models that we were capable of utilizing correctly and effectively was small. For example, the effectiveness of neural networks diminishes when applied to small datasets (otherwise, the model tends to overfit and perform poorly when testing). This meant that applying models that are capable of dealing with small datasets were optimal.

Additionally, because of the high proportion of True labels in our data set (approximately 60% of the data set were true statements), learning to differentiate between truth and lies (let alone two types of lies) was extremely difficult. Lastly, because the data was divided according to interviewer and interviewee statements, when attempt-

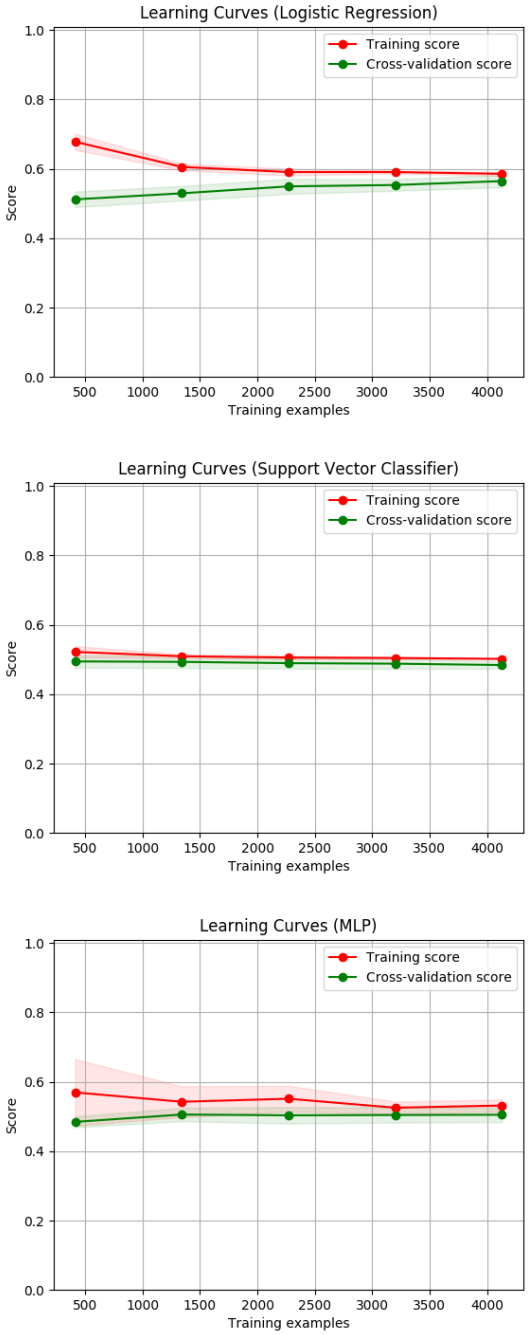

Figure 4: Learning curves for our logistic regression, support vector and multi-layer perceptron classifiers. These learning curves were generated using acoustic, prosodic and lexical features, with variance thresholding as well as random over sampling to have a 50:50 class split. Observing the trend of these learning curves, the cross validation scores for all of them converge relatively low at around 0.6; thus, more data will most likely not improve these models' performance.

ing to analyzing the interviewees responses, our training examples were predominately filled with silence (corresponding with the interviewers question). This made reasonable feature construction extremely difficult, seeing as the features that we did obtain from featurizers like openSMILE software suite were potentially biased towards silence.

6.2 Future Directions

Potential future research directions include further feature construction, especially lexical features using the provided transcripts. One possible interesting route for feature engineering could be to somehow analyze the interviewees' responses relative to the question presented by the interviewer (i.e. attempt to capture the interview context). Additionally, the inclusion of additional signals and data can be helpful. As shown with the performance of bagging and boosting, more data can potentially help these models and providing additional signals can potentially help other learning models differentiate between classes. Additional data would also potentially help remedy the skew that is present in the data set in its current form. While we decided to analyze lies at the word level, we could also potentially try to analyze the data at the phone level, thus producing a much larger data set, which could help neural network performance. Lastly, further hyperparameter tuning for the current models can always be helpful.

Acknowledgments

We would like to thank the CS 224S staff and instructor, Andrew Maas, for offering the opportunity to work on this project; additionally, we would like to thank the Stanford Department of Linguistics and Sebastian Schuster for access to the CSC corpus.

References

Frank Enos, Elizabeth Shriberg, Martin Graciarena, Julia Hirschberg, and Andreas Stolcke. 2007. Detecting deception using critical segments http://www.cs.columbia.edu/ frank/papers/enos_et_al_is2007.pdf.

Florian Eyben, Felix Weninger, Florian Gross, and Bjrn Schuller. 2013. Recent developments in opensmile, the munich open-source multimedia feature extractor. *ACM Multimedia* pages 835–838. http://audeering.com/technology/opensmile/.

Martin Graciarena, Elizabeth Shriberg, Andreas Stolcke, Frank Enos, Julia Hirschberg, and Sachin Kajarekar. 2006. Combining prosodic lexical and cepstral systems for deceptive speech detection. *IEEE International Conference on Acoustics, Speech and Signal Processing* http://www.cs.columbia.edu/nlp/papers/2006/graciarena_al_06.pdf.

Julia Hirschberg, Stefan Benus, Jason M. Brenier, Frank Enos, Sarah Friedman, Sarah Gilman, Cynthia Girand, Martin Graciarena, Andreas Kathol, Laura Michaelis, Bryan Pellom, Elizabeth Shriberg, and Andreas Stolcke. 2005. Distinguishing deceptive from non-deceptive speech. *INTERSPEECH* pages 1833–1836. http://www.cs.columbia.edu/ julia/files/hirschberg_al_05.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780. http://www.bioinf.jku.at/publications/older/2604.pdf.

Andrew Maas. 2017. Lecture 11: Emotion/affect extraction and interpersonal stance http://web.stanford.edu/class/cs224s/lectures/224s.17.lec13.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Verónica Pérez-Rosas, Rada Mihalcea, Alexis Narvaez, and Mihai Burzo. 2014. A multimodal dataset for deception detection. *Language Resource Evaluation Conference* pages 3118–3122. http://www.lrec-conf.org/proceedings/lrec2014/pdf/869_Paper.pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, and Yoshua Bengio. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf.

Juan F. Torres, Elliot Moore, and Ernest Bryant. 2008. A study of glottal waveform features for deceptive speech classification. *IEEE International Conference on Acoustics, Speech and Signal Processing* http://ieeexplore.ieee.org/document/4518653/.