Deception Detection in Online Mafia Game Interactions

Lisa Fu
Department of Computer Science
Stanford University
Stanford, CA 94304
lfu2@stanford.edu

March 18, 2019

Abstract

In this project, we use deep learning to detect deception through online conversations in the interaction-based party game Mafia. We present various neural network frameworks (LSTM, bidirectional LSTM, CNN) to generate predictions of deceptive text. The results of this paper suggest the capability of neural network architectures to effectively detect deception in the Mafia setting, and offer qualitative analysis in agreement with current deception research findings.

1 Introduction

We all lie, for a variety of reasons. Sometimes, they can be little white lies, maybe to flatter a recipient or protect a child. In more devastating cases, they can lead to the innocent on death row, or result in evasion of criminal cases dealing with murder or national security.

Historically, humans have devised countless methods to try to detect deception, whether for personal, business, or judicial affairs. From closely watching someone's body behavior for micro expressions and nervous gestures, to consulting oracles or psychics, to polygraph tests, humans have invested a lot of energy and effort to understanding deception [1].

Yet, differentiating the truthful from the deceptive proves a tricky task, as there is no single behavior that can clearly signal or indicate deception. Research suggests that even law enforcement agents (such as from the FBI, CIA, Drug Enforcement Agency) trained to read body cues for deception don't do much better than chance in distinguishing the truth [2].

Detecting deception using NLP has gained traction in recent years, and we predict that there are linguistic differences in the behavior between deceivers and truthtellers, and that those subtle nuances may be captured through various

deep learning methods. In this project, we hope to develop a system that is sensitive and robust enough to identify the reflections of deceptive language. Due to the limited amount of deception corpuses available and the elusive nature of the problem, we will be scoping our problem definition to detecting deception in the interaction-based strategy game Mafia. The findings of this project will help to inform future research in emerging areas of linguistics and deep learning, with many potential applications such as for criminal justice, eliminating cultural and racial bias, or detecting fake news across various media platforms.

2 Background and Related Work

2.0.1 The Game of Mafia

The original face-to-face game of Mafia, also known as Werewolf, consists of interactions surrounding a conflict between two groups: an informed minority (the mafia), and the uninformed majority (the innocents). At the start of the game, each player is assigned a secret identity affiliated with either the mafia or the innocents. The game starts with an in-game "Day", where players discuss their suspicions of the identities of Mafia members, while the Mafia members pretend to agree and try to influence the votes in their favor without arousing more suspicion. The game is played out in two alternating phases, where the mafia may covertly "murder" an innocent, and where surviving players debate the secret identities of the mafia and vote to eliminate a suspect. The game ends when all of the mafia have been eliminated, or until the mafia outnumber the innocents [4].

Because of the popularity of the game, it has been adopted by a large number of online communities, most notably Mafiascum, and played out through public forum threads or chat rooms.

2.0.2 Mafia as a Model for Deception

Due to lessened pressure and unnaturalness felt in a game environment, Mafia offers a promising venue to study deception in a real-life setting. Zhou and wei Sung (2008) collected 1192 Mafia games from a popular Chinese website to explore deceptive cues in group dynamics and found that the deceivers' average word count to be lower than that of the truthtellers'. Ruiter and Kachergis (2018) scraped Mafiascum's Normal Game archives to create a publicly available dataset of player conversations. They hand-picked linguistic features based on prior deception research and a set of average word vectors enriched with subword information that correlated with truthful or deceptive roles, such as word count per 24 hours, ratio of third/second/first-person pronouns, and ratio of negative emotion words. An SVM classifier fit on a combination of the feature sets achieved an area under the precision-recall curve of 0.35 (chance = 0.26), and an AUROC of 0.64 (chance = 0.50). According to their findings, only 2 features (sentence length and ratio of third-person pronouns) were directly backed up by

primary meta-analysis research, suggesting that the strength of linguistic cues is highly context-dependent [5].

In this paper, we describe our attempt to use neural models (LSTM, Bidirectional LSTM, CNN) for detecting deception within the Mafia game context.

3 Methodology

3.1 Dataset

3.1.1 Data

Our primary dataset is the Mafiascum dataset, a large scale source of deceptive text collected from over 700 games of Mafia on an internet forum (Mafiascum) by Ruiter and Kachergis [3]. The dataset contains over 9000 documents, each containing messages written by a single player in a single game.

3.1.2 Preprocessing

A lot of preprocessing was handled by Ruiter and Kachergis, such as discarding all conversations after a game has already ended, or discarding games that were not completely aligned. For every player in every game, the remaining in-game posts were concatenated into a single text document, and assigned a label signifying whether the player was Mafia or not.

We sanitized the dataset further to remove documents containing invalid characters, and removed documents with less than 50 words as recommended by Ruiter and Kachergis (2018), since document word length might be a confounding factor in the dataset.

3.2 Models

3.2.1 Model 1: Baseline

Bag of Words + Logistic Regression For our baseline, we performed a binary classification using simple logistic regression with scikitlearn (C=1.0, solver=lbfgs). The documents were first transformed via bag of words model into one-hot encoded vector representation, then applied through logistic regression using 10-fold cross validation.

3.2.2 Model 2: Recurrent Neural Networks

LSTM Our baseline was able to achieve relatively good performance, but we wanted to explore whether factoring in positional information could further improve performance. We used a Recurrent Neural Network (RNN) architecture that incorporated unidirectional Long Short-Term Memory (LSTM) cells to counter the problems of vanishing and exploding gradients, using the implementation from [6].

Given our matrix of labeled Mafiascum player documents, at each timestep we apply a dropout (p=0.5) on the input layer to prevent neurons from over-fitting, then provide the row of documents as input to the LSTM.

Then, the aggregated output probabilities of the model are fed into a softmax layer to compute the softmax cross-entropy loss between the predicted labels and true labels. We impose a L2 regularization constraint on the weights, and use the Adam optimization algorithm to minimize the loss.

Bidirectional LSTM Furthermore, we extend our RNN LSTM model to a bidirectional LSTM model, using the implementation from [6]. Bidirectionality allows our model to access to future and past contexts (rather than just a linear interpretation) by introducing a second layer, where hidden connections flow in an opposite temporal order [8]. We apply a dropout (p=0.5) on the input layers, and the forward and backward outputs are concatenated together before being passed on to the next layer. Similar to the previous RNN LSTM architecture, the outputs are fed into a softmax layer, with L2 regularization and Adam optimization algorithm to determine the best weights for each epoch to minimize the loss.

3.2.3 Model 3: Convolutional Neural Network

Lastly, we train a simple Convolutional Neural Network (CNN) as outlined by Yoon [7] using the implementation from [6]. At a high level, the first layer is an embedding layer, which maps vocabulary word indices into low-dimensional vectors.

The next convolutional layers produce convolutions using filters of different sizes (we used filter sizes of 3, 4, and 5 words at a time). Each convolution produces tensors of different shapes, and thus all have separate layers, and the results across the filters are maxpooled into a single feature vector.

The results are fed through a dropout layer (p=0.5 during training) to prevent neurons from co-adapting, and a softmax layer to minimize the softmax cross-entropy loss [7].

4 Experiments

4.1 Dataset

For our deception detection model, we used the Mafiascum dataset [3]. This dataset contains over 9000 documents, with a single document pertaining to all of the online conversations/posts of a single person in a single game. After preprocessing the dataset (and reducing documents due to memory limitations), there were 3500 labeled documents, with 2800 documents for training, and 700 documents randomly selected for testing. The document word counts ranged from 50 to 11,000+ words, with most within the range of 500-3000 words. The documents were padded to be equal to the max length of all documents, and the

vocabulary was extracted from the documents using Tensorflow VocabularyProcessor to map each word to an index, before feeding the vector of indices into the various models [6].

4.2 Evaluation Method

We compare the predictions against the labels provided by the Mafiascum dataset. For each of the models, we calculate accuracy, precision, recall, F_1, and AUROC (Area Under Receiver Operating Characteristic Curve) scores. The algorithms for accuracy, precision, recall, and F_1 are defined as follows:

$$Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$$
$$Precision = \frac{tp}{tp+fp}$$
$$Recall = \frac{tp}{tp+tn}$$
$$F_1 = \frac{2 \cdot precision \cdot recall}{precision+recall}$$

4.3 Experimental Details

We ran all neural models for 50 epochs at first, with batch size of 32. The model was saved as a checkpoint and cross validated on a dev set (10% of the train set) every 100 steps. The best model across the checkpoint was used to evaluate the test set. Most hyperparameters were chosen based on default values.

Hyperparameters for all models:

- Dropout probability: 0.5
- Learning rate for Adam optimizer: 0.001
- L2 regularization lambda: 0.001

Hyperparameters for LSTM and Bi-LSTM:

- Hidden size: 128
- Number of layers: 2

Hyperparameters for CNN:

- Embedding size: 256
- Filter sizes: 3, 4, and 5
- Number of filters: 128

4.4 Results

The test performances of the various experiments are displayed in Table 1:

Model	AUROC	Accuracy	Precision	Recall	$\mathbf{F}_1$
Logistic Regression	0.816	0.816	0.815	0.820	0.815
Unidirectional LSTM	0.880	0.888	0.874	0.880	0.875
Bidirectional LSTM	0.945	0.947	0.948	0.945	0.946
CNN	0.948	0.949	0.948	0.948	0.948

Table 1

4.4.1 RNN Model

Figure 2 shows the training loss and accuracy curves for the unidirectional LSTM (top) and bidirectional LSTM (bottom). The unidirectional LSTM started overfitting after 1,000 steps (around 12 epochs), so the checkpoint model saved closest to 1,000 steps was used to evaluate the test set. In the future, more finetuning of the hyperparameters might be necessary to obtain more accurate results.

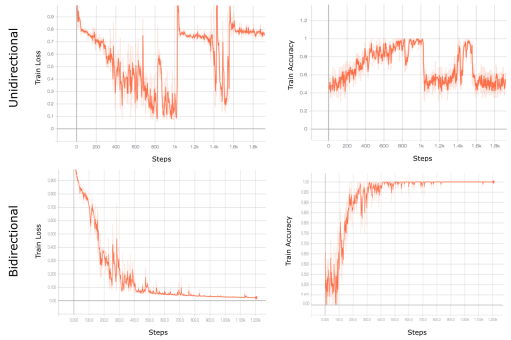

Figure 1: Loss/Accuracy Curves for LSTM/Bi-LSTM

4.4.2 CNN Model

Figure 2 shows the loss/accuracy curves for the CNN model.

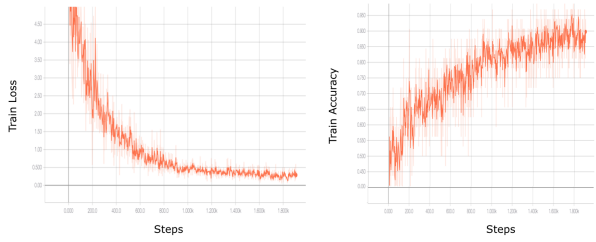

Figure 2: Loss/Accuracy Curves for CNN

5 Analysis

5.1 Qualitative Analysis

After evaluating the test set with the trained RNN models, we computed term frequencies for a subset of 20 randomly selected documents for each neural architecture from each category: mafia (prediction) / mafia (actual), innocent / mafia, innocent / innocent, and mafia / innocent. Below are a few example word clouds of the top 30 words for the Bi-LSTM model:

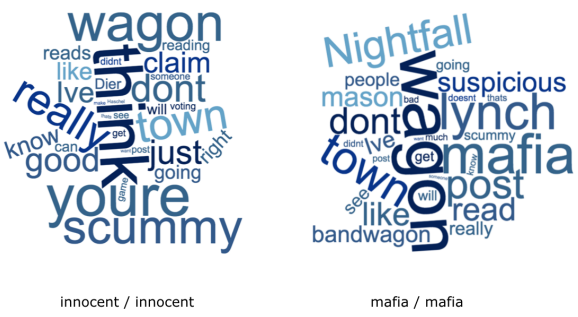

Figure 3: Word Clouds for Bi-LSTM Correct Labels

Semantics of "lynch" more commonly associated with mafia identity In Mafia, a "Lynch" is the town's collective killing method, where innocents vote which members to "lynch" and mafia try to influence innocents to not "lynch" them. All neural models (LSTM/Bi-LSTM/CNN) correctly predicted mafia identities when the word "lynch" was in the top 15 word frequencies, and conversely correctly identified innocent identities when "lynch" was not in the top 15 word frequencies. This suggests a few ideas, that the intentionally deceptive people were more aware of their imminent "lynching" and tried to take a more active approach in influencing the innocent. Another possibility could be that innocent people had less pressure to prove they were innocent, and thus needed to use related terminology less.

Role of First, Second, and Third Person Pronouns Both LSTM and Bi-LSTM models identified innocent players correctly when first person/exclusive words were more frequent, such as "youre" and "Ive". This aligns with deception research findings that liars tend to use less first-person pronouns to disassociate themselves from their lie [9]. However, it is impossible to determine whether this is influenced by deceptive behavior, or if there is some other confounding variable, such as cultural or socioeconomic differences.

Positive/Negative Emotions Current deception research findings also suggest that deceptive language contains more negative emotion [9]. Both the bidirectional RNN-LSTM and CNN models were able to capture this in their term frequencies. The words "good" and "right" surfaced to the top for correctly identified innocent players across the Bi-LSTM and CNN models, whereas the word "suspicious" became prominent for the Bi-LSTM results.

6 Conclusion and Future Improvements

We have trained tensorflow frameworks based on unidirectional LSTM, bidirectional LSTM, as well as CNN, that successfully perform deception detection on online Mafia game interactions using the Mafiascum dataset [3]. All three models performed above baseline accuracy (81.6%), with 88.8% (unidirectional LSTM), 94.7% (bidirectional LSTM), and 94.9% (CNN). This suggests that deep learning techniques may be able to effectively identify linguistic cues that signal deception.

However, there may be many confounding variables such as culture, socioeconomic status, and personal beliefs that may influence a person's language, rather than intent of deceit. Furthermore, the models used in this project may not be generalizeable for out of domain situations. Because the game of Mafia is still a simulated environment, the linguistic cues could be especially overt compared to real life scenarios, where it is not guaranteed that a subset of people within a group dynamic will always play a "deceitful" role.

Given the time and machine constraints for this project, many future explorations include:

- Transform dataset to paragraph vectors or reduce dimensionality of document text
- Implement self-attention with LSTM neural architecture to increase contextual awareness
- Instead of choosing only 20 documents to compute term frequency analysis, go through entire corpus for more precise analysis

Acknowledgments

I'd like to thank Annie Hu for providing insightful feedback for the initial project direction, and for the CS224N teaching staff for being so timely and helpful across all of my interactions!

References

[1] "The End of Detecting Deception." Psychology Today, Sussex Publishers.

[2] Adelson, Rachel. “Detecting Deception.” Monitor on Psychology, American Psychological Association, www.apa.org/monitor/julaug04/detecting.

[3] de Ruiter, Bob Kachergis, George. (2018). The Mafiascum Dataset: A Large Text Corpus for Deception Detection.

[4] “Mafia (Party Game).” Wikipedia, Wikimedia Foundation, 4 Mar. 2019.

[5] Lina Zhou and Yu wei Sung. 2008. Cues to deception in online chinese groups. In Proceedings of the 41st Hawaii International Conference on System Sciences. pages 146–46.

[6] Liu, Haoyou. “Zackhy/TextClassification.” GitHub, 15 July 2018, github.com/zackhy/TextClassification.

[7] Britz, Denny. “Implementing a CNN for Text Classification in TensorFlow.” WildML, 5 Feb. 2016, www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/.

[8] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H. Xu, B. (2016). Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. ACL.

[9] Dou, J., Liu, M., Muneer, H. Schlussel, A. (2015). What Words Do We Use to Lie?: Word Choice in Deceptive Messages. CHI.