Corpus-based Deception Detection using Neural Models

Tom Grigg[*]
MSc CSML
19151291

Ritwick Sundar[*]
MSc ML
19125221

Kelvin Ng[*]
MSc ML
707184

Nicolas Ward[*]
MSc ML
19133564

Abstract

The aim of this project is to investigate the use of neural models for deception detection. We evaluate the performance of a number of classes of neural models in this domain by observing their ability to identify the underlying roles of players in the deceptive game of Mafia (also referred to as "Werewolf"). To this end, we make use of the Mafiascum dataset, a collection of over 700 text-based games of Mafia played in online forums, where players were randomly assigned either deceptive (Mafiosi) or non-deceptive (Villager) roles. We demonstrate that neural models are able to match or even outperform models that utilise hand-crafted stylometric features derived from extensive psychological and linguistic studies by domain experts, as in the Mafiascum baseline established in de Ruiter and Kachergis (2018).

1 Introduction

Automatic deception detection is a challenging problem in natural language processing, which has numerous legal, political, and social applications. Deception is a complex human behaviour, the manifestations of which depend on non-textual context, intrinsic properties of the deceiver (e.g. education, linguistic competence, etc.), and the specific nature of the deceptive act (e.g. playing a game with friends vs lying in court). Text-based deception detection is therefore a much more nuanced and difficult problem than more traditional text classification benchmarks such as sentiment analysis and topic identification. Practical difficulties of deception detection that are not present in these simpler tasks include the fact that human performance on the task is often at chance, and that the signal in available data is dim and potentially sparse. From a statistical perspective, acts of deception are implicitly adversarial, corresponding to an attempt to appear to belong to the distribution of non-deceivers. This is especially the case in the game of Mafia, where it is in a deceiver's interest to explicitly pretend to be a villager.

Deception detection systems are desirable in many sectors of society; numerous related studies trying to find relevant signals have taken place, particularly within the fields of psychology and linguistics (Feng et al., 2012a; Juola, 2012). However, due to the extreme variation in the contexts of deception, creating effective deception detection systems is a challenging exercise in feature engineering, with practitioners having to start from scratch in each new context. Indeed, the original Mafiascum paper notes that their model found certain hand-picked features to be predictive, but in the opposite direction to that suggested by the existing literature.

We believe that deep architectures should be able to automatically learn and detect the hidden signals relevant for deception detection in a given context, thus reducing reliance on expert domain knowledge and overall effort required to develop such systems. In this project, we investigate the use of neural models for deception detection. Due to the limited amount of deception corpora available, and the elusive nature of the problem, we approach this challenge in the context of detecting deception in the interaction-based strategy game of Mafia. Ultimately, our aim is to explore the ability of neural models to identify the underlying roles of players in the game (which is assimilated to the broader task of text classification in adversarial contexts).

The findings of this project could contribute to the development of future research in emerging areas of linguistics and deep learning, with many promising applications ranging from measuring the authenticity of online reviews, detecting

[*]Equal contribution. Author ordering determined by coin flip over a Zoom conference.

fake news across media platforms, and assessing the reliability of statements made in criminal trials based on tribunal hearing transcripts.

2 Background and Related Work

Until the 1970s, research aimed at detecting deception was largely focused on finding nonverbal cues (body language and facial expressions) for use in face-to-face interactions. After Mehrabian (1971) found that slow and sparse speech can indicate deception, systemic research into linguistic cues to deception took off.

Burgoon et al. (2003) found that the multivariate analysis of indicators of complexity at both the sentence level (simple sentences, long sentences, short sentences, sentence complexity, number of conjunctions, average words per sentence) and vocabulary level (vocabulary complexity, number of big words, average syllables per word) did not produce overall multivariate effects, but several individual variables did show the effects of deception. They showed that deceivers had significantly fewer long sentences, average words per sentence and sentence complexity compared to truth tellers. This meant that their language was less complex and easier to understand.

In a recent meta-analysis of 79 linguistic cues to deception from 44 studies, Hauch et al. (2015) found that, relative to truth-tellers, deceivers experienced greater cognitive load, expressed more negative emotions, distanced themselves more from events by expressing fewer sensory and perceptual words, and referred less often to cognitive processes.

These works suggest that quantifiable latent signals correlated with the hidden deception variable do exist, so that a sufficiently powerful model might be able to quantify and use them to identify deception. In particular, they demonstrate the effectiveness of features derived from text analysis, which frequently include basic linguistic representations such as n-grams and sentence count statistics (Mihalcea and Strapparava, 2009; Ott et al., 2011) and also more complex linguistic features derived from syntactic context free grammar trees and part of speech tags (Feng et al., 2012b).

2.1 The Game of Mafia

The face-to-face game of Mafia (also referred to as “Werewolf”) is a social deduction game which models a conflict between two groups: an informed minority (the mafia), and the uninformed majority (the villagers). At the start of every game, each player is assigned a secret identity affiliated with one of these two groups. The game has two alternating phases: Day and Night. During the game time ‘Day’, players debate the identities of all involved and vote to eliminate a suspect; at ‘Night’, the Mafia covertly kill members of the innocents group. The game continues until one group achieves it’s winning condition: for the innocents, this means eliminating all Mafia members, while for the Mafia, this means reaching numerical parity with the innocents.

Due to the popularity of the game, it has been adopted by a large number of online communities, with the game being played out through public forum threads or chat rooms. In addition to its entertaining dynamics, the reason for our interest in the game of Mafia is that, by design, members of the Mafia group have a strong incentive to conceal their identity through deception.

2.2 The Mafiascum Dataset

The Mafiascum dataset is a large scale source of deceptive text collected from over 700 games of Mafia on the Mafiascum internet forum, compiled by de Ruiter and Kachergis as part of the original Mafiascum paper. The dataset consists of over 9000 documents, each containing the concatenated messages of a single player in a single game.

2.3 Deception Detection in Mafiascum

The original Mafiascum study by de Ruiter and Kachergis (2018) demonstrates that a linear model utilising traditional linguistic stylometric features does not generalise across all contexts. The authors extract such traditional features from the meta-analysis by Hauch et al. (2015) and show that few of them are predictive in the context of Mafia. Furthermore, of the six that are predictive (ratio of 3rd-person pronouns, ratio of 2nd-person pronouns, “but” ratio, sentence length, use of sensory words, ratio of quantifiers), they establish that only two of them are predictive in the direction expected by the meta-analysis (sentence length and ratio of 3rd-person pronouns), suggesting that the strength and direction of linguistic cues is highly context dependent (Zhou and Sung, 2008). They proceed to train a simple logistic regression model using these predictive features. They then take two different sets of semantic embeddings, FastText and GloVe, tokenize and embed each document in the dataset via its mean token embedding, and plug this embedding into another logistic regression, which achieves a small yet significant performance increase over the hand-picked stylometric model. The authors then combine the hand-picked feature set with the mean semantic embeddings to train a final logistic regression which again significantly outperforms either individual feature set, acting as evidence that embedding the documents has produced new predictive information, likely specific to the domain of Mafiascum. We interpret these results as significant evidence for our beliefs that neural models are capable of autonomously extracting features relevant to deception detection. Throughout this work, our aim is to build upon de Ruiter and Kachergis's approach by training custom neural architectures (implementations of LSTM, CNN & BERT) to detect deceptive Mafia players. These more complex neural models have considerable advantages of non-linearity, and we expect these models to intercept more nuanced cues by accounting for context, co-references and variable semantic relationships.

3 Methodology

Here, we formalise our approach to the deception detection problem in online Mafia, and present the modified dataset we used, along with an overview of the models that were implemented.

3.1 Dataset Modification (DailyMafiascum)

We noticed early on that the 10,000 documents contained in the original Mafiascum dataset were not sufficient to properly train our neural networks. Our models would either underfit or overfit on the data, and were extremely difficult to regularize. From the results of our debugging experiments, we suspect that this was due to both the sparsity and complexity of latent signals in the data, combined with the length of the documents, which makes propagating gradient information difficult, especially when there is only a small amount of signal to begin with. We therefore modified the original Mafiascum dataset by redefining the notion of a document: due to the cyclic nature of a game of Mafia, we surmised that there would more often than not be significant information in any single day of activity for a given user in a given game. As a result, we modified the code for the original Mafiascum dataset to create labelled documents corresponding to a single user in a single game, in a single twenty-four hour period of activity. This resulted in a dataset of over 100,000 documents. This dataset, which we refer to as DailyMafiascum, was certainly noisier than the original Mafiascum dataset, but contains useful information segmented into more reasonable chunks that allow the models to train more effectively. We were careful to eliminate confounding factors in the DailyMafiascum dataset, the most obvious of which was that a given user in a given game should not have one game-day belonging to the training set, and another game-day in the validation set, since these datapoints would essentially share a label and the neural models could be capable of memorising an individual user's idiosyncratic behaviours, rather than detecting deceptive behaviour. This was the only suspected confounding factor that appeared to be an issue in our tests, and a simple but satisfactory solution was to ensure that the training, validation and testing sets are disjoint with respect to the users they contain.

3.2 Data Preprocessing

Since we made use of the Mafiascum dataset, a large part of the preprocessing was already handled by de Ruiter and Kachergis. These preprocessing steps were carried forward to our DailyMafiascum dataset, including discarding all post-elimination ("Twilight") and post-game discussions where users are more likely to freely announce their roles. Documents below 50 words in total were discarded in the original Mafiascum dataset, and we elected to do the same in DailyMafiascum. We additionally found a small number of duplicate datapoints in the original dataset, which we removed in our DailyMafiascum dataset. We also decided to handle player replacement in exactly the same way as in the original dataset, whereby if a player is explicitly replaced in a game due to another player quitting, the two players' documents are kept separate.

3.3 Models

Each model architecture consists of a "feature construction" component and a final classification component. The final classification component in all of our models is a single fully-connected linear layer, where each individual neuron corresponds to a single dimension of what can naturally be interpreted as feature space. This feature space is hand-crafted in the case of the "hand-picked" Logistic Regression described below, and defined/learned by the neural model in all other cases.

Logistic Regression Since we train our neural models on the larger DailyMafiascum dataset, our results are not necessarily directly comparable to the results obtained in de Ruiter and Kachergis (2018) on the original Mafiascum dataset, though we expect them to be similar. We therefore implement a Logistic Regression model with the same parameters, but train it on DailyMafiascum for use as a baseline. The features used in the regression model are kept the same as in the original Mafiascum paper, but each feature value is recomputed over the daily documents in the DailyMafiascum dataset. We train three separate logistic regression models for comparison: one on the hand-picked features alone, the second on the mean FastText embedding of the words in the document, and the final model on the combined feature set. This model's loss is weighted by the log of the length of the document sample, in order to encourage it to learn from the less noisy document.

Convolutional Neural Network (CNN) Their ability to reduce the size of text vectors, coupled with their ability to extract features, make CNNs good building blocks for our deception detection task. CNNs are applied to embedding vectors of a given sentence in the hope that they will manage to extract useful features (such as phrases and relationships between words that are closer together in the sentence). The convolution layers help reduce the dimensionality of the text and act as summaries of sorts which are then fed to a series of dense layers. These summaries capture the relationships and meaning of the sentences, and are useful for classification.
In the following, we briefly go through the different layers in our CNN architecture. Our CNN is parameterized by a convolution layer with kernel size k and $\mathcal{F}$ filters, a ReLU activation function $g(x) = \max(0, x)$, a dropout function D_p which zeroes its input column-wise with probability p, a max pooling operation and a final sigmoid classification layer $\sigma : \mathbb{R}_2^{\mathcal{H}} \to (0, 1)$. Given a document $X \in \mathbb{R}^{e \times n}$ where n is the length of the document and e is the size of the token embeddings, our model f can be decomposed as follows:

$$\begin{aligned}
H(X) &= CONV_{\mathcal{F},k}(X) \in \mathbb{R}^{n \times \mathcal{H}_1(k)} \\
A(X) &= \max(0, H(X)) \in \mathbb{R}^{n \times \mathcal{H}_1(k)} \\
M(X) &= \text{maxpool}(A(X)) \in \mathbb{R}^{\mathcal{H}_1(k)} \\
O(X) &= D_p(M(X)) \in \mathbb{R}^{\mathcal{H}_1(k)} \\
f(X) &= \sigma(O(X)) \in \mathbb{R}
\end{aligned}$$

The final values for these parameters can be seen in our final model table 4.

Long Short-Term Memory (LSTM) Our implementation of LSTM is parameterized by two hidden layer sizes $\mathcal{H}_1$ and $\mathcal{H}_2$, a dropout function D_p which zeroes its input column-wise with a specified probability p, a boolean flag for bidirectionality $\mathcal{B} \in \{\texttt{True}, \texttt{False}\}$ and a pooling operation $\text{pool} \in \{\texttt{mean}, \texttt{max}\}$, and a final sigmoid classification layer $\sigma : \mathbb{R}_2^{\mathcal{H}} \to (0, 1)$. Given a single document $X \in \mathbb{R}^{e \times n}$ where n is the length of the document and e is the size of the token embeddings, our model f corresponds to the following composition:

$$\begin{aligned}
H_1(X) &= D_p(\text{LSTM}_{(\mathcal{H}_1,\mathcal{B})}(X)) \in \mathbb{R}^{n \times \mathcal{H}_1} \\
H_2(X) &= \text{LSTM}_{(\mathcal{H}_2,\mathcal{B})}(X) \in \mathbb{R}^{n \times \mathcal{H}_2} \\
O(X) &= \text{pool}(H_2) \in \mathbb{R}^{\mathcal{H}_2} \\
f(X) &= \sigma(O(X))
\end{aligned}$$

The final values for these parameters can be seen in the appendix.

Bidirectional Encoder Representations from Transformers (BERT) Language model pretraining has shown to be effective in a number of natural language processing tasks. Among these models, the Google-released BERT (Devlin et al., 2018) is a conceptually simple, but empirically powerful, model that uses the transformer architecture (Vaswani et al., 2017).

Since the same pre-trained model successfully tackles a broad set of NLP tasks, we had reason to believe that it would work well on our task in detecting deception. The task-specific models can be formed by incorporating the pre-trained BERT model with one additional output layer, so a minimal number of parameters need to be learned from scratch. This process is known as fine-tuning.

Following the fine-tuning procedure detailed in (Devlin et al., 2018), they provided possible hyperparameter values that work well across all tasks. We decided to use a conservative value of 16 as the Batch Size, 2e-5 as our Learning rate (Adam), along with a linear decay. The model ran for 4 epochs.

Our model is based on the implementation by Wolf et al. (2019). It is built upon the base BERT DeepPavlov/bert-base-cased-conversational model (Burtsev et al., 2018), which has 12 transformer layers, 110 Million parameters, 12 self-attention heads, and a hidden size of 768. We decided to use this as our pre-trained model as it was trained on the English versions of Twitter, Reddit, DailyDialogues and OpenSubtitles, which presumably contain more deceptive content.

3.3.1 BERT for sequence classification (BERT + CLS)

This is the BERT model for sequence classification mentioned in (Devlin et al., 2018). The token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. This is fed into a fully connected layer with *tanh* activation, and then another layer with output dimension 2 before the softmax as shown in Figure 1.

The pre-trained model available mostly has a limit of 512 tokens, including 2 special tokens at the beginning and end. Assuming conclusion tends to be at the end, we take the last 510 tokens of each documents.

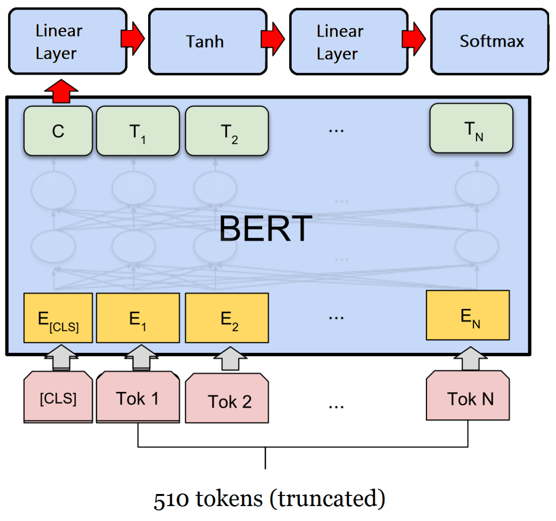

Figure 1: BERT + CLS fine tuning architecture on top of the pre-trained BERT model. Image from (Devlin et al., 2018)

3.3.2 BERT for sequence classification with MIn-MaX Pooling (BERT + MIXCLS)

In order to overcome the limitation of BERT pre-train model accepting no more than 512 tokens, we propose a new model on top of the existing BERT + CLS model. We feed a sequence of multiple N tokens with overlapping stride as shown in Figure 2. We then aggregate the multiple output of the hidden state corresponding to the CLS token after the linear layer and the *tanh* activation by concatenating $[min; max]$. We found the optimal value of N is around 256.

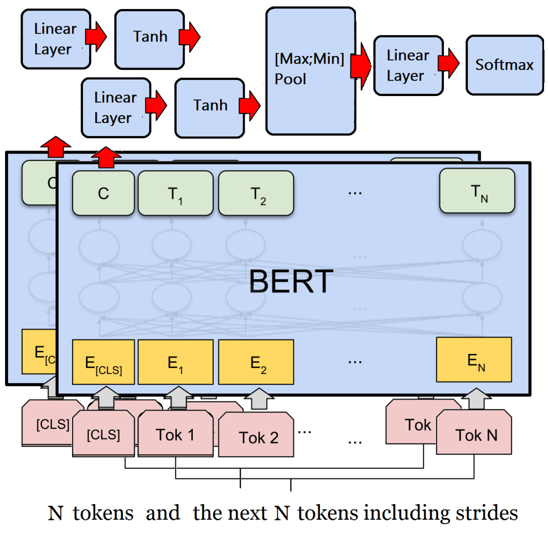

Figure 2: BERT + MIXCLS fine tuning architecture on top of the pre-trained BERT model. Image from (Devlin et al., 2018)

Due to limited hardware resources, we demonstrated this model with 4x256 tokens. Given more resources, we would like to investigate the use of a larger BERT model.

4 Experiments

4.1 Evaluation metrics

It is important for a deception detection model to be both sensitive and specific. In the context of Mafia in particular, this balance is especially important, since it is arguably as bad to falsely accuse a Villager as it is to fail to detect a Mafiosi. To this end, we ultimately evaluate eachmodel by its average precision (AP) and its area under the Receiver Operator Characteristic curve (AUROC/AUC).

AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\mathrm{AP} = \sum_n (R_n - R_{n-1}) P_n$$

where P_n and R_n are the precision and recall at the n-th threshold. Note that P_n and R_n are defined as:

$$P_n = \frac{TP}{TP + FP}$$
$$R_n = \frac{TP}{TP + FN}$$

with TP, FP and FN the numbers of true positives, false positives and false negatives predicted by the model, respectively.

The AUC score captures the discriminative power of a model. It can be interpreted as the model's ability to separate the two class distributions, i.e. the probability that a randomly chosen positive example is ranked above a randomly chosen negative example. Essentially, the AUC score allows us to filter out models that are representative but not discriminative: we do not care about accuracy since the majority of players are not Mafia.

4.2 Training Procedure

Optimization Each of our neural models was trained via ADAM optimization using the mean binary cross entropy as a loss function, i.e. for a given model f and a batch of examples $B = (x_i, y_i)_{i \in 1 \ldots n}$ we compute:

$$-\frac{1}{n} \sum_{1}^{n} [y_i \log(f(x_i)) + (1 - y_i) \log(1 - f(x_i))]$$

where $y_i = 1$ if x_i represents a Mafiosi, and $y_i = 0$ otherwise. We use the cross entropy loss since encouraging the model to move towards the correct class in probability is desirable, as even if it does not explicitly flip the label over the threshold, it will increase its discriminative power and hence the AUC and AP. While training model, the two classes were reweighted appropriately in proportion to their respective sizes, as a considerable imbalance of the two classes was observed (approximately 24-76 Mafia-Villager).

Training, Validation, Testing Our models take quite a long time to train due to their relative complexity, the large size of our DailyMafiascum dataset, and our limited resources. We therefore elected to forego the 20-fold cross validation recommended in the original Mafiascum paper, and split our data into a 50-50 train-test split, followed by a 70-30 train-validation split. The large size of our test set allows us to be confident in each of our final models' metric values, while also, from a pragmatic perspective, reduces the time required to train models to run our experiments. The relatively large size of our validation set allowed us to be more confident that our model would generalize well to the unseen data in the test set.

4.3 Tuning

Model parameter and hyperparameter optimization was done initially via a wide but relatively sparse grid search, followed by a round of manual tuning, which was then followed by a number of more finely grained grid searches for the Logistic Regression, CNN and LSTM. Our BERT model's training time was far too long for us to be able to perform extensive tuning - based on the observed loss curves, we expect that with more computational resources the BERT model would easily be able to outperform the others with larger BERT models.

5 Results

Model	**AUC**	**AP**
Log. Reg. (HP)	0.535	0.261
Log. Reg. (FastText)	0.527	0.258
Log. Reg. (HP + FastText)	0.541	0.265
B-LSTM (GloVe)	**0.596**	**0.299**
CNN (GloVe)	0.570	0.282
BERT + CLS	0.553	0.278
BERT + MIXCLS	**0.578**	**0.297**

Table 1: Final Test Performance of the implemented models trained on the DailyMafiascum dataset.

5.1 Analysis

From Table 1, it is clear that B-LSTM with mean-pooling across history and BERT for sequence classification with Min-Max Pooling are our best-performing models. We have therefore clearly demonstrated that neural models are able to implicitly engineer features that allow them to outperform hand-picked stylometric features derived from the aforementioned studies.

5.2 Convergence of Stylometric Features

While it is clear that our neural models perform much better than the Logistic Regression models that make use of the stylometric features, it is also certainly true that by modifying the dataset to increase the number of datapoints and reduce the length of the documents, we have degraded the amount of information contained in the stylometric feature values because we have computed them for each user's postings over a twenty-four hour period, rather than over the entirety of a game. While there is an argument to be made for the fact that this means that our neural models are able to perform better given less information in a single datapoint, it is also not the purpose of our analysis. Hence, Table 2 gives the values of the Logistic Regression trained on hand-picked features that have been allowed to take on their values from the original Mafiascum dataset, i.e. they have been computed over the entirety of the game for each user and set for each datapoint corresponding to that user's game-days. The baseline models do benefit from allowing these "global" stylometric features to converge, but they still perform significantly worse than our best neural models trained on the locally computed version of the dataset. We felt this was important to showcase to aid comparison with the original Mafiascum results.

Model	**AUC**	**AP**
Log. Reg. (HP)	0.564	0.283
Log. Reg. (FastText)	0.527	0.258
Log. Reg. (HP + FastText)	0.568	0.284

Table 2: Final Test Performance of the implemented models trained with hand-picked feature values computed over the entirety of each game rather than the 24-hour period.

6 Discussion

From our experiments, it is clear that neural models are capable of automatically learning representations that are most likely better than traditional linguistic features for a specific context - at least better than those reported in Hauch et al. (2015). Our best performing model was a fairly simple 2-layer unidirectional LSTM which mean-pooled across the history of its hidden representations. We actually implemented this model first, inspired by the mean-pooling Logistic Regression of the original Mafiascum paper. We expected the CNN to outperform the LSTM because we surmised that the CNN, by the nature of its architecture, would be better at learning to represent syntactic and stylometric features; we expected such representations to be more effective since the hand-picked stylometric Logistic Regression significantly outperformed the mean-pooled semantic FastText embedding Logistic Regression in our baseline experiments. It is interesting to observe that this was not the case, and it perhaps agrees with the intuition that, although syntax and stylometric features are effective simple indicators for deception, the most effective (and possibly human) way to tell if someone is purposefully trying to deceive is to semantically understand their words in the context that they are presented and come to a decision based on semantic experience. Perhaps the power of the LSTM comes from its greater ability to read the semantic embeddings in order. Regardless of whether or not this anthropomorphism of our models holds any semblance of truth, it certainly warrants further investigation into state-of-the-art language understanding models on the task of deception detection.

Model	**AUC**	**AP**
B-LSTM (GloVe)	**0.945**	**0.946**
CNN (GloVe)	0.931	0.934
BERT + MIXCLS	0.976	0.978

Table 3: Final Test Performance of the implemented models trained on the IMDB Sentiment Analysis dataset.

It is important to note that there are many other psychological studies on deception and potentially many other stylometric indicators for deception that neither our study nor the original Mafiascum paper have investigated. It is almost certainly the case that a team of Mafiascum regulars would have little trouble deceiving our best model and would find it much harder to deceive a domain expert in the form of another regular. However, the key tradeoff is in the time it would take to become a Mafiascum expert, versus the time it takes to train a neural network to recognise statistical patterns in the data. While it is demonstrably true that effective representations can be learned for deception detection in the niche domain of text-based online Mafia, there are many other contexts of deception that remain unexplored due to the lack of data. In certain contexts deception data acquisition is very difficult, since there is limited opportunity for ”real world” data. Deceivers do not typically reveal their labels, and if a deceiver is discovered, then they are evidently not a very good one! Indeed, we found model tuning in this domain to be magnitudes more difficult than in more typical text classification tasks. In fact, we decided to test our architectures on a more traditional dataset of a similar size: the IMDB sentiment analysis dataset, and were easily able to achieve AUC scores of 0.9 and above (see Table 3).

References

Judee K. Burgoon, J. Pete Blair, Tiantian Qin, and Jay F. Nunamaker. 2003. Detecting deception through linguistic analysis. In *International Conference on Intelligence and Security Informatics*, pages 91–101. Springer, Berlin, Heidelberg.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. 2018. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012a. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012b. Distributional footprints of deceptive product reviews. In *Sixth International AAAI Conference on Weblogs and Social Media*.

Valerie Hauch, Iris Blandón-Gitlin, Jaume Masip, and Siegfried L. Sporer. 2015. Are computers effective lie detectors? A meta-analysis of linguistic cues to deception. *Personality and social psychology Review*, *19*(4):307–342.

Patrick Juola. 2012. Detecting stylistic deception. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 91–96. Association for Computational Linguistics.

Albert Mehrabian. 1971. Nonverbal betrayal of feeling. *Journal of Experimental Research in Personality*.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312. Association for Computational Linguistics.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 309–319. Association for Computational Linguistics.

Bob de Ruiter and George Kachergis. 2018. The Mafiascum Dataset: A Large Text Corpus for Deception Detection. *arXiv preprint arXiv:1811.07851*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Lina Zhou and Yu W. Sung. 2008. Cues to deception in online Chinese groups. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, page 146. IEEE.

Appendix

Below we present our best hyperparameters for each of our neural models. Together with the formalisations in Section 3, these define our models.

Params.	**LSTM**	**CNN**	**BERT**
Learning rate	0.0001	0.0001	0.00002
Weight decay	0.00015	0.025	N/A
Dropout	0.55	0.6	0.5
Bidirectional	False	N/A	N/A
Pooling	mean	max	[min; max]
Filters	N/A	300	N/A

Table 4: Best parameters for each implemented model.