Deception Detection: Lie Detection in Diplomacy via Machine Learning

Grishma Bellani (2022189) Palak Bhardwaj (2022344) Piyush Narula (2022354)

Abstract

The research investigates online relationship deception patterns by utilizing information from "It Takes Two to Lie: One to Lie, and One to Listen" (Peskov et al., 2020). It presents a novel approach to deception detection in diplomatic communication using deep learning models. We analyzed messages exchanged in diplomatic settings to identify deceptive content. This study aims to enhance understanding of trust and betrayal behavior in communication.

1. Introduction

1.1. Problem Definition

Deception detection in the game of Diplomacy involves strategies where players can negotiate, be diplomatic, or betray. This project utilizes the 2020_acl_diplomacy dataset (Peskov et al., 2020) to build machine learning models for lie detection in game messages. The dataset comprises 17,289 annotated messages. The research aims to create a model that integrates deception indicators (specific to the domain) through deep learning models combined with natural language processing techniques. This study motivates us to develop an intense system that employs deep learning methods alongside progressed natural language processing for identifying specific domain deception signals within multi-agent dialogues.

1.2. Motivation

Deception Detection is an important aspect of various different kinds of fields. Detect enables crime prevention and creates better communication while increasing system and human trust. In this Project, we explore complex features like power dynamics and alliance chances and how they detect deception detection. In addition to this, Security & Safety function as two applications of deception detection technology since the method identifies deception to protect borders and enforce laws and defend against cyber threats and dating scams.

Humans focus on behavioral deception detection through the study of why people lie coupled with deception identification techniques which leads to research on emotional and cognitive mechanisms, hence this is a very relevant topic for discussion.

2. Dataset Description

The dataset includes 17,289 messages derived from 12 Diplomacy games, with approximately 5% marked as fraudulent. Each message includes sender and receiver identification, providing truth status and deception suspicion rankings ("true," "false," or "None").
Key elements include:
1.Message text
2.Game score (ranging from -18 to 18)
3.Number of supply centers (0 to 18)
4.Score delta
5.Temporal markers (season and year)
Messages are identified using absolute and relative indexes, game IDs, and the communicating parties. The dataset is split into three subsets:
1.train.jsonl (9 out of 12 games)
2.validation.jsonl (1 game)
3.test.jsonl (2 game)

`Dataset Link`

3. Baseline Model

The primary baseline model is **Context LSTM + Power**. It achieves a macro F1 score of 55.13 for actual lies, while the **Context LSTM** alone reaches 53.7. These serve as effective baselines. The LSTM model specifically achieves a macro F1 of 53.7 and a Lie F1 (Actual Lie) of 12.58. Additional models explored include: 1.Bagofwords.py
2.harbingers.py
3.humanbaseline.py
4.andomandmajoritybaselines.py

We also experimented with the TF-IDF approach and LSTM variants. As mentioned in the original paper, even BERT and LSTM variants did not yield performance improvements.

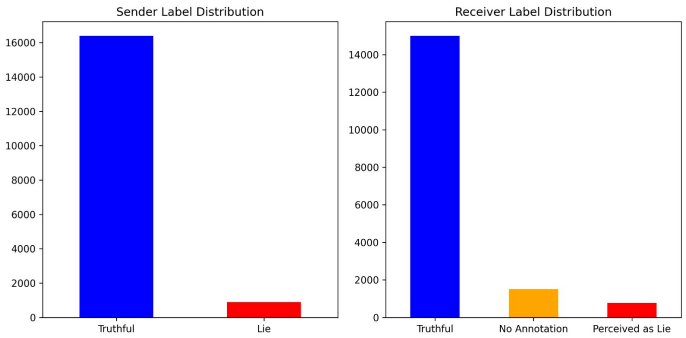

Figure 1. Model performance or system workflow.

4. Methodology

Our primary objective was to handle the class imbalance in the deception detection dataset available at 2020_acl_diplomacy Github Repository.

4.1. Data Preprocessing

We preprocessed our data to convert raw conversational data into a more structured input format for the model. The current dataset $\mathcal{D} = \{G_1, G_2, \ldots, G_N\}$ consists of games, where each game G_i is a tuple:

$$G_i = (\mathbf{m}_i, \mathbf{s}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{y}_i, \mathbf{g}_i, \delta_i, \mathbf{l}_i^s, \mathbf{l}_i^r),$$

with:

- $\mathbf{m}_i = [m_{i1}, \ldots, m_{iT_i}]$: Sequence of T_i messages (strings).
- $\mathbf{s}_i, \mathbf{r}_i \in \mathcal{C}^{T_i}$: Speaker/receiver countries.
- $\mathbf{t}_i \in \mathcal{T}^{T_i}, \mathbf{y}_i \in \mathcal{Y}^{T_i}$: Seasons/years.
- $\mathbf{g}_i, \delta_i \in \mathbb{R}^{T_i}$: Game scores/deltas.
- $\mathbf{l}_i^s, \mathbf{l}_i^r \in \{-1, 0, 1\}^{T_i}$: Sender/receiver labels (1 = deceptive, 0 = truthful, -1 = unannotated).

Empty messages are set to "[EMPTY]".

4.1.1 Categorical Mapping

Categorical variables are mapped to integer indices using dictionaries computed on the training set:

- **Countries**: $\mathcal{C} \to \{0, 1, \ldots, N_c - 1\}$, padding index N_c.
- **Seasons**: $\mathcal{T} \to \{0, 1, \ldots, N_t - 1\}$, padding N_t.
- **Years**: $\mathcal{Y} \to \{0, 1, \ldots, N_y - 1\}$, padding N_y.

For a country $c \in \mathcal{C}$, the mapping is:

$$\text{Map}_c(c) = k, \quad k \in \{0, \ldots, N_c - 1\}.$$

These mappings enable embedding layers to represent categories as dense vectors.

4.1.2 Linguistic Feature Extraction

For each message $m_{ij} \neq$ "[EMPTY]", six linguistic features are extracted using spaCy for tokenization and part-of-speech tagging, and VADER for sentiment analysis. Let W_{ij} be the tokenized words of m_{ij}. The features are:

1. **Word Count**: Log-transformed length:

$$f_{ij1} = \log(1 + \min(|W_{ij}|, 100)).$$

2. **Lexical Diversity**: Unique words ratio:

$$f_{ij2} = \min\left(\frac{|\text{set}(W_{ij})|}{|W_{ij}| + 10^{-8}}, 1.0\right).$$

3. **Self-References**: Count of pronouns (e.g., "I", "me"):

$$f_{ij3} = \min\left(\sum_{w \in W_{ij}} \mathbb{1}_{w \in \{\text{"I"}, \text{"me"}, \ldots\}}, 10\right).$$

4. **Group References**: Count of collective pronouns (e.g., "we", "us"):

$$f_{ij4} = \min\left(\sum_{w \in W_{ij}} \mathbb{1}_{w \in \{\text{"we"}, \text{"us"}, \ldots\}}, 10\right).$$

5. **Negative Emotion**: VADER's negative score:

$$f_{ij5} = \text{VADER}(m_{ij})_{\text{neg}}.$$

6. **Modifiers**: Count of adjectives/adverbs:

$$f_{ij6} = \min\left(\sum_{w \in W_{ij}} \mathbb{1}_{\text{pos}(w) \in \{\text{ADJ, ADV}\}}, 10\right).$$

Feature Normalization For "[EMPTY]" messages, $\mathbf{f}_{ij} = \mathbf{0} \in \mathbb{R}^6$. Features are standardized using training set statistics:

$$\mu_f = \frac{1}{N_f} \sum_{i,j} \mathbf{f}_{ij}, \quad \sigma_f = \sqrt{\frac{1}{N_f} \sum_{i,j} (\mathbf{f}_{ij} - \mu_f)^2} + 10^{-8},$$

$$\mathbf{f}_{ij} \leftarrow \frac{\mathbf{f}_{ij} - \mu_f}{\sigma_f}.$$

The feature tensor is $\mathbf{F}_i \in \mathbb{R}^{T_i \times 6}$.

4.1.3 Output Representation

Each conversation is a dictionary:

$$\{\mathbf{m}_i, \mathbf{F}_i, \mathbf{s}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{y}_i, \mathbf{g}_i, \delta_i, \mathbf{l}_i^s, \mathbf{l}_i^r\},$$

where $\mathbf{s}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{y}_i$ are mapped indices, and all sequences are padded to length $T_{\max}$ within a batch using:

$$\text{pad}(\mathbf{x}_i, T_{\max}, v) = [\mathbf{x}_i; v, \ldots, v] \in \mathbb{R}^{T_{\max}},$$

with padding values $v = N_c$ for countries, $v = 0$ for others.

4.1.4 Class-Imbalance Handling

1.Oversampling Deceptive Conversations
To address class imbalance, conversations with at least one deceptive label are oversampled:

$$\mathcal{D}' = \mathcal{D} \cup \bigcup_{k=1}^{K-1} \{G_i \mid \exists j : l_{ij}^s = 1\},$$

with $K = 2$, increasing the representation of deceptive instances.
2. Weighted Loss
the loss function is weighted to emphasize more on lie(which are only 5% in the dataset)

4.2. Data batching + Padding

Conversations can have multiple lengths hence dynamic padded was implmented by us.

4.3. Model Architecture: EnhancedDeceptionDetector

The `EnhancedDeceptionDetector` is a neural network integrating text, linguistic, and contextual features. For a batch of B conversations with maximum length $T_{\max}$, inputs are:

$$\{\mathbf{m}_b, \mathbf{F}_b, \mathbf{s}_b, \mathbf{r}_b, \mathbf{t}_b, \mathbf{y}_b, \mathbf{g}_b, \delta_b, T_b\}_{b=1}^{B}.$$

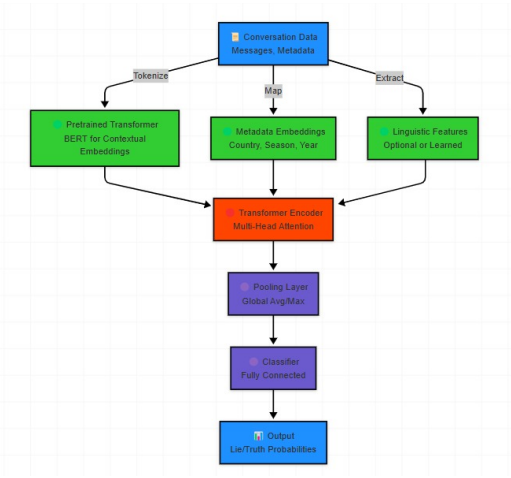

Figure 2. Preprocessing

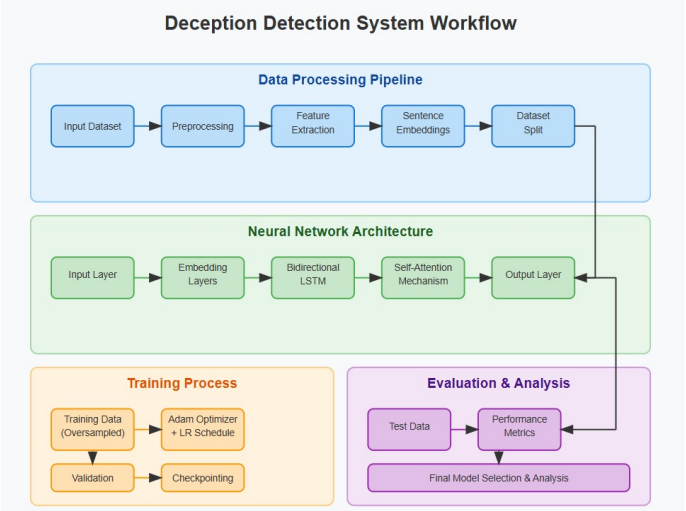

Figure 3. WorkFlow

4.3.1 Embedding Layers

- **Sentence Embeddings**: Sentence-BERT (paraphrase-MiniLM-L6-v2) encodes messages:

  $\mathbf{e}_{bj} = \text{Sentence-BERT}(m_{bj}) \in \mathbb{R}^{384}$, $\mathbf{e}_{bj} = \mathbf{0}$ if $m_{bj} =$ "[EMPTY]".

  Padded tensor: $\mathbf{E}_b \in \mathbb{R}^{T_{\max} \times 384}$.

- **Country Embeddings**: Embedding layers map indices:

  $\mathbf{C}^s_{bj} = \text{Emb}_c(s_{bj})$, $\mathbf{C}^r_{bj} = \text{Emb}_c(r_{bj})$, $\text{Emb}_c : \{0, \ldots, N_c\} \to \mathbb{R}^{16}$.

  Output: $\mathbf{C}^s_b, \mathbf{C}^r_b \in \mathbb{R}^{T_{\max} \times 16}$.

- **Season/Year Embeddings**:

  $$\mathbf{T}_{bj} = \text{Emb}_t(t_{bj}), \quad \mathbf{Y}_{bj} = \text{Emb}_y(y_{bj}),$$

  where $\text{Emb}_t : \{0, \ldots, N_t\} \to \mathbb{R}^8$, $\text{Emb}_y : \{0, \ldots, N_y\} \to \mathbb{R}^8$. Output: $\mathbf{T}_b, \mathbf{Y}_b \in \mathbb{R}^{T_{\max} \times 8}$.

- **Linguistic Features**: Projected features:

  $$\mathbf{F}'_{bj} = \text{Linear}_f(\mathbf{F}_{bj}), \quad \text{Linear}_f : \mathbb{R}^6 \to \mathbb{R}^6,$$

  yielding $\mathbf{F}'_b \in \mathbb{R}^{T_{\max} \times 6}$.

4.3.2 Feature Integration

Concatenate features:

$\mathbf{X}_b = [\mathbf{E}_b, \mathbf{F}'_b, \mathbf{C}^s_b, \mathbf{C}^r_b, \mathbf{T}_b, \mathbf{Y}_b, \mathbf{g}_b, \delta_b] \in \mathbb{R}^{T_{\max} \times (384+6+16+16+8+8+1+1)}$,

where $\mathbf{g}_b, \delta_b \in \mathbb{R}^{T_{\max} \times 1}$. Project to hidden space:

$$\mathbf{H}_b = \text{Linear}_p(\mathbf{X}_b) \in \mathbb{R}^{T_{\max} \times 64}, \quad \text{Linear}_p : \mathbb{R}^{440} \to \mathbb{R}^{64}.$$

Apply dropout with probability $p = 0.4$:

$$\mathbf{H}_b \leftarrow \text{Dropout}(\mathbf{H}_b, p = 0.4).$$

4.3.3 Sequence Modeling

A bidirectional LSTM processes the sequence:

$$\mathbf{O}_b, (\mathbf{h}_b, \mathbf{c}_b) = \text{LSTM}(\text{Pack}(\mathbf{H}_b, T_b)),$$

where Pack uses lengths T_b to handle variable lengths, and $\text{LSTM} : \mathbb{R}^{64} \to \mathbb{R}^{128}$ (64 per direction). Output: $\mathbf{O}_b \in \mathbb{R}^{T_{\max} \times 128}$. Normalize:

$$\mathbf{O}_b \leftarrow \text{LayerNorm}(\mathbf{O}_b).$$

4.3.4 Self-Attention Mechanism

Compute attention weights:

$$\mathbf{a}_b = \text{softmax}(\text{Linear}_a(\mathbf{O}_b)) \in \mathbb{R}^{T_{\max} \times 1}, \quad \text{Linear}_a : \mathbb{R}^{128} \to \mathbb{R}^1,$$

$$a_{bj} = \frac{\exp(\text{Linear}_a(\mathbf{O}_{bj}))}{\sum_{k=1}^{T_{\max}} \exp(\text{Linear}_a(\mathbf{O}_{bk}))}.$$

Form context vector:

$$\mathbf{c}_b = \sum_{j=1}^{T_{\max}} a_{bj} \cdot \mathbf{O}_{bj} \in \mathbb{R}^{128}.$$

Normalize: $\mathbf{c}_b \leftarrow \text{LayerNorm}(\mathbf{c}_b)$.

4.3.5 Classification

Predict logits:

$$\mathbf{z}_b = \text{Linear}_c(\mathbf{c}_b) \in \mathbb{R}^1, \quad \text{Linear}_c : \mathbb{R}^{128} \to \mathbb{R}^1.$$

Broadcast to all messages:

$$\mathbf{Z}_b = \text{expand}(\mathbf{z}_b, T_{\max}) \in \mathbb{R}^{T_{\max} \times 1}.$$

4.3.6 Loss Function

Use binary cross-entropy with logits, weighted for class imbalance:

$$w_p = 1.5 \cdot \frac{N_0}{N_1},$$

where N_0, N_1 are the counts of truthful and deceptive labels. The loss is:

$$\mathcal{L} = \frac{1}{\sum_{b,j} \mathbb{1}_{l^s_{bj} \neq -1}} \sum_{b,j : l^s_{bj} \neq -1} \text{BCE}(\mathbf{z}_{bj}, l^s_{bj}; w_p),$$

$$\text{BCE}(z, l; w) = -[w \cdot l \log(\sigma(z)) + (1 - l) \log(1 - \sigma(z))],$$

with $\sigma(z) = \frac{1}{1+e^{-z}}$ and a mask excluding $l^s_{bj} = -1$.

4.4. Training Procedure

Training optimizes the model over the dataset $\mathcal{D}'$.

4.4.1 Optimization

- **Optimizer**: Adam with weight decay $\lambda = 0.01$.
- **Learning Rate Scheduler**: Linear warm-up for 10% of steps:

$$\eta_t = \eta \cdot \min\left(\frac{t}{T_w}, 1.0\right),$$

where $T_w = 0.1 \cdot E \cdot \lceil |\mathcal{D}'|/B \rceil$, $E = 15$ epochs, B is batch size.

4.4.2 Training Loop

Training Loop epoch = 1 to E Shuffle $\mathcal{D}'$ batch $\mathcal{B} = \{G_b\}_{b=1}^{B}$ in $\mathcal{D}'$ Pad inputs to $T_{\max} = \max_b T_b$ Compute $\mathbf{Z}_{\mathcal{B}} = \text{Model}(\mathcal{B})$ Compute $\mathcal{L}$ with mask $l_{bj}^s \neq -1$ Backpropagate $\nabla\mathcal{L}$ Update parameters via Adam Step scheduler Evaluate on validation set $\mathcal{D}_{\text{val}}$ Save checkpoint if macro F1 > 0.55 or False F1 > 0.18

4.4.3 Checkpointing

Save model state, optimizer, scheduler, and metrics when validation metrics meet thresholds. The best model maximizes macro F1, with False F1 as a tiebreaker, saved as:

best$_c heckpoint.pt$best$_c heckpoint.pt$best$_c heckpoint.pt$best$_c heckpoint.pt.$

4.5. Evaluation Metrics

Evaluate predictions $\hat{y}_{bj} = \mathbb{1}_{\mathbf{z}_{bj}>0}$. Metrics include:

- **Per-Class** (for classes $c \in \{0, 1\}$):

$$\text{Prec}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}, \quad \text{Rec}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c},$$

$$\text{F1}_c = \frac{2 \cdot \text{Prec}_c \cdot \text{Rec}_c}{\text{Prec}_c + \text{Rec}_c}.$$

- **Micro-Averaged**:

$$\text{Prec}_{\text{micro}} = \frac{\sum_c \text{TP}_c}{\sum_c (\text{TP}_c + \text{FP}_c)},$$

$$\text{Rec}_{\text{micro}} = \frac{\sum_c \text{TP}_c}{\sum_c (\text{TP}_c + \text{FN}_c)}.$$

- **Macro-Averaged**:

$$\text{Prec}_{\text{macro}} = \frac{\text{Prec}_0 + \text{Prec}_1}{2}, \quad \text{F1}_{\text{macro}} = \frac{\text{F1}_0 + \text{F1}_1}{2}.$$

- **Confusion Matrix**: $\mathbf{M} \in \mathbb{Z}^{2\times 2}$, where M_{ij} counts true class i predicted as j.

4.6. Hyperparameter Tuning

Perform grid search over:

$$\text{lr} \in \{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}\}, \quad B \in \{4, 8\}.$$

For each combination, train a model and select parameters maximizing validation macro F1:

$$(\text{lr}^*, B^*) = \arg\max_{\text{lr},B} \text{F1}_{\text{macro}}(\mathcal{D}_{\text{val}}).$$

4.7. Analaysis of all methods

We tried multiple other approaches taking inspiration from the intial models and reference articles we explored. We came up with Approaches like :

- Dual LSTM
- Roberta Implementation with Fusion Mechansim
- Multiheaded Attention with Bi-LSTM
- Use of Feedback Loop
- Simple LSTM with all feature Integration
- Bi-LSTM + Positional Enocding

As mentioned in the github link - href{ https://github.com/palak-b19/Deception-Detection

5. Experimental Setup

5.1. Model Parameters

- Sentence Embedding from Sentence-BERT: Dimension = 384
- Linguistic Features Dimension: 6
- Country Embedding Dimension: 16
- Season Embedding Dimension: 8
- Year Embedding Dimension: 8
- Hidden Dimension: 64
- Dropout Rate: 0.4

5.2. Training Parameters Used

- Grid search for hyperparameter tuning
- Learning rates tested: [0.00005, 0.0001, 0.0002]
- Batch sizes tested: [4, 8]
- Number of epochs: 15
- Early stopping with patience
- Class weight adjustment: Positive class weighted by $(\frac{\text{num-truths}}{\text{num-lies}}) \times 1.5$

5.3. Implementation Details

- Framework used: PyTorch
- Text processing libraries: SpaCy, NLTK
- Sentiment Analysis: VADER
- Sentence Embedding: Sentence-BERT

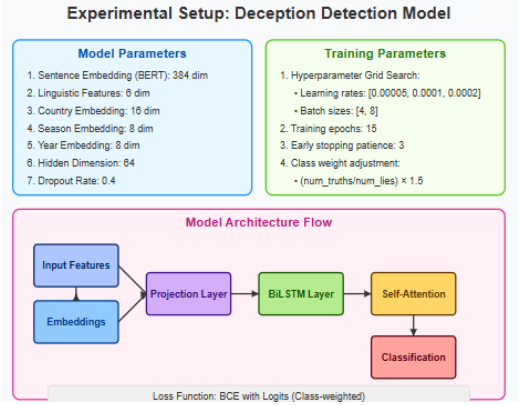

Figure 4.

5.4. Auxiliary Techniques

- **Seed Setting**: Fix seeds for reproducibility:

$$\text{seed} = 100.$$

- **Device Management**: Use CUDA if available, else CPU.
- **Logging**: Record progress and timing.
- **Checkpoint Backup**: Zip checkpoints for portability.

6. Related Work

A deception detection model developed within this work uses a bidirectional LSTM alongside Sentence-BERT embeddings and linguistic features which include word count, lexical diversity, self references and sentiment scores for text message deception classification. The established design of the model draws from important research articles which explore deep learning systems and text-based deception feature development approaches.

1. **Hybrid Acoustic-Lexical Deep Learning Approach for Deception Detection (Mendels et al., 2017)**. Published at Interspeech 2017 and available at `https://www.semanticscholar.org/paper/fda72ce8b95866dd924326b09fca35a4a68ecab7`, Mendels et al. developed a combination method using deep learning for deception detection assessment that operated on the Columbia X-Cultural Deception (CXD) Corpus. A neural network classifier trained to detect truthful and deceptive speech receives input from acoustic features and lexical features like word frequency data and syntactic patterns as well as indicators of sentiment.

   Linguistic patterns present in transcriptions provided effective input for the detection system through feature extraction of lexical components. The model's text-based component uses Sentence-BERT to create semantic message embeddings which mirror the semantic strengths of analytical lexical features modeled in their research. The developed embeddings receive additional enhancement from linguistic features including calculated word counts calculated as:

$$f_{\text{word count}} = \log(1 + |\text{words}|), \tag{1}$$

   and the term "I" or "me" indicates a self-reference which is counted during frequency analysis. A bidirectional LSTM processes these features because they use sequential processing similar to their neural network to predict deception labels. The procedure of building lexical feature vectors adopted from Mendels et al. follows this sequence:

$$\mathbf{f}_l = [f_1, f_2, \ldots, f_n], \quad f_i = \phi(w_i, \text{context}), \tag{2}$$

   A text sample receives features through $\mathbf{f}_l$ while the function ϕ extracts features from word w_i together with its context. We also compute lexical diversity:

$$f_{\text{lexical diversity}} = \frac{|\text{unique words}|}{|\text{words}| + \epsilon}, \quad \epsilon = 10^{-8}, \tag{3}$$

   The fusion of Sentence-BERT embeddings with Sentence-B occurs before the input to the LSTM system. The neural classifier techniques used by the authors guide our binary classification model design by applying binary cross-entropy loss with logits:

$$\mathcal{L}(y, x) = \max(x, 0) - x \cdot y + \log(1 + \exp(-|x|)), \tag{4}$$

   where the model logit output (x) serves as the deception indicator variable together with the deception label is $y \in \{0, 1\}$.The model receives a positive weighting to manage class distribution because the researchers sought optimal performance on uneven deception datasets.

2. **Deception Detection Using Machine Learning and Deep Learning Techniques: A Systematic Review (Prome et al., 2024)**. Published in *Neurocomputing* and accessible at `https://www.sciencedirect.com/science/article/pii/S2949719124000050`, Prome et al. provide a thorough review of deception detection methodologies, deep learning models especially the LSTMs demonstrate exceptional effectiveness when processing text-based applications.

   The analysis evaluates research that uses neural networks to analyze conversations by stressing the value of linguistic components derived from instruments such as LIWC or sentiment analyzers which extract pronouns and negative expressions and emotional tone. The research solution works to solve the fundamental problem of unbalanced deceptive data by recommending weighted loss techniques for rare deceptive detection.

   This model builds on reported findings by using a Sentence-BERT model with bidirectional LSTM architecture to process message series while determining semantic features as recommended for text analytics. The analysis incorporates LIWC-inspired linguistic indicators by measuring both negative emotional intensity through VADER sentiment analysis and counting modifiers matching adjectives and adverbs according to the review's suggested evidence of deception.

   The LSTM's sequential processing is formalized as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{5}$$
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{6}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{7}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{8}$$
$$h_t = o_t \cdot \tanh(c_t), \tag{9}$$

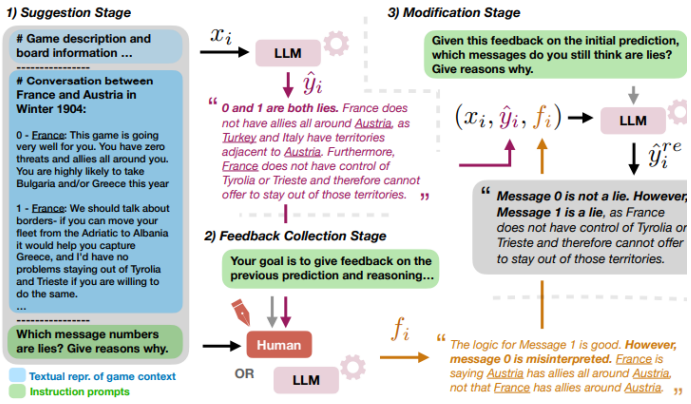

Figure 5. Enter Caption

This model comprises the input parameters x_t that consist of embeddings together with linguistic features while integrating h_{t-1} from the previous hidden state with learnable parameters W_* and b_* alongside activation functions σ and $\tanh$. The bi-directional LSTM network analyzes forward and backward sequence context according to Prome et al. research to enhance detection of deceptive text. Weighted loss function serves as the implementation for addressing class imbalance after reviewing the paper's suggested approach. To address class imbalance, we implement a weighted loss function, inspired by the review's recommendations:

$$\mathcal{L}_{\text{weighted}} = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot \mathcal{L}(y_i, x_i), \quad w_i = \begin{cases} \text{pos\_weight} & \text{if } y_i = 1 \\ 1 & \text{if } y_i = 0 \end{cases} \tag{10}$$

where $\text{pos\_weight} = 1.5 \cdot \frac{\#\text{ truths}}{\#\text{ lies}}$ is dynamically adjusted based on recall, ensuring focus on deceptive messages.

3. **LLMs are Superior Feedback Providers: Bootstrapping Reasoning for Lie Detection with Self-Generated Feedback (Banerjee et al., 2024)**. Published in *arXiv* and accessible at `https://arxiv.org/abs/2408.13915`, A new bootstrapping infrastructure based on large language models (LLMs) allows Banerjee et al. to improve Diplomacy game conversation lie detection without requiring training data.

   A three-stage procedure constitutes this method starting with suggestion where GPT-3 or similar affordable LLM produces initial lie predictions followed by feedback collection when LLMs or human experts evaluate these predictions; and modification, where a more advanced LLM (e.g., GPT-4) refines outputs using feedback.

   Not like the previous work, this work does not takes into account the true label, which leads to a 39% improvement in lying-F1 score over zero-shot baselines and using the supervised LSTM models.

   This study is uses LLM's generated feedback as this is 13 times cheaper and 29% more efficient in lie-F1. The Diplomacy dialogues provide linguistic cues which are under analysis using contextual embeddings while the feedback system detects both systematic errors and false negative cases. Sequential prediction models and their framework structure can be expressed as probabilistic sequence tasks:

$$\hat{y}_i \sim P_{\mathcal{M}}(\cdot \mid x_i), \quad f_i \sim P_{\mathcal{M}}(\cdot \mid x_i, \hat{y}_i, p_f), \quad y_i^{re} \sim P_{\mathcal{M}}(\cdot \mid x_i, \hat{y}_i, f \tag{11}$$

where x_i represents game state and dialogue inputs, $\hat{y}_i$ are initial predictions, f_i is feedback, and y_i^{re} are refined predictions, with prompts p_f and p_m guiding feedback and modification stages. To address class imbalance in lie detection, a weighted scoring mechanism is implied, prioritizing rare deceptive messages, similar to:

$$\mathcal{L}_{\text{weighted}} = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot \mathcal{L}(y_i, \hat{y}_i), \quad w_i = \begin{cases} \text{pos\_weight} & \text{if } y_i = 1, \\ 1 & \text{if } y_i = 0, \end{cases} \tag{12}$$

where pos_weight is adjusted to emphasize lies, aligning with the paper's focus on improving recall for deceptive messages in imbalanced datasets.

7. Discussion and Observations

The neural network system equipped with self-attention mechanisms performs deception detection in strategic communication texts. Key components include:

The pre-trained model generates sentence embeddings Six linguistic features (word count, lexical diversity, self-references, etc.)

A record of contextual metadata provides information about the game scores together with the playing seasons and nations involved. LSTM with bidirectional processing and self-attention.

A weighted loss function alongside oversampling serves to optimize the model for handling class imbalance during training and assessment of model performance contains macro-F1 scores as well as False class detection (lies).

During gameplay when players focus on scoring data the author presents a novel approach to the link between deception and strategy tactics. What makes alliances along with message timing unimportant?

In case of ties during checkpoints False F1 (truths) are employed to rank predictions despite their unexplained preference for true messages as opposed to false ones. The 10% learning rate warmup for LSTMs is a subtle enhancement used to promote stability which the authors have kept hidden from plain sight.

8. Limitations

- **High-Class Imbalance:** With only 5% of the data representing lies, the dataset is highly imbalanced. Although oversampling is used, duplicating deceptive conversations may lead to overfitting and reduce the diversity of training data.
- **Limited Context Understanding:** The model lacks deeper understanding of strategic relationships and historical communication patterns between players.
- **Binary Classification (True/Lie):** The deception is treated as a binary problem, even though deceptive behavior exists on a continuum with varying degrees of subtlety.
- **Lack of Interpretability:** While self-attention mechanisms improve performance, they reduce the model's interpretability, making it difficult to explain the predictions.

- **Training Efficiency:** The model requires substantial computational resources due to repeated training and checkpoint saving.

- **Model Architecture Limitations:** LSTM-based models struggle with long sequences due to gradient vanishing issues. The fixed dimension size (384) in Sentence-BERT also limits adaptability. Supporting a wider range of embedding models could improve flexibility.

- **Domain Specificity:** The model is tailored for deception in the Diplomacy game domain and lacks transferability to other real-world deception contexts such as interviews or social media.

9. Possibilities of exploration and Future Work

- Automated Feature Extraction: The neural network should acquire linguistic features instead of requiring manual construction of them. The implementation of CNNs or RNNs can extract features from basic text elements. The use of learned features shows improvement in generalizability because they recognize subtle cues which manual design methods cannot identify.

- Hierarchical Modeling: A hierarchical network must model dialogues at three levels (message and turn and game) to track context at several different levels of detail. The use of hierarchical modeling becomes necessary since deception appears in varying formats between different scales.

- Interpretability Analysis: A SHAP or LIME analysis should be implemented to determine which characteristics like words, speakers and sentiment most affect deception prediction results. The model's decision-making process can be analyzed for both trust building purposes and debugging activities.

10. Conclusion

The EnhancedDeceptionDetector model functions as a well-designed solution that incorporates Sentence-BERT semantic embeddings with spaCy and VADER linguistic features and national data and historical metadata elements to detect deception in diplomacy game text. Its strengths include:

- Multimodal Approach: Combines text, linguistics, and game metadata for a holistic view of deception.

- The model improves performance and addresses unbalanced data through weighted loss techniques together with parameter optimization methods and oversampling techniques.

- Complete Evaluation: that shows multiple measurement types, including class-level and micro/macro results, and stores checkpoints for repeatable testing

- Modular Design: for distinct processing, model building and learning procedures, yet its dependencies on manually designed attributes along with basic attention mechanisms and domain-specific framework assumptions need additional development.

- The preprocessing system demonstrates thoroughness yet requires automated system integration and adaptation to larger use.

- The model demonstrates satisfactory results according to reported F1-scores but its ability to work across different domains has not been tested.

Since NLP and DL techniques are being used, and they are proved to be a good combination for deception detection systems. Directed enhancements would enable this system to analyze dishonest communication through multiple scenarios.

11. Results

Method	Accuracy (%)	Macro F1 Score	Lie F1 Score
Bi-LSTM Approach	81.75	57.71	24.60
LLM Feedback Loop	69.19	53.51	26.51

Table 1. Comparison of Bi-LSTM and LLM Feedback Loop methods

Method	Accuracy (%)
Bi-LSTM Approach	81.75
LLM Feedback Loop	69.19

Table 2. Accuracy Comparison of Bi-LSTM and LLM Feedback Loop methods

Method	Macro F1 Score
Bi-LSTM Approach	57.71
LLM Feedback Loop	53.51

Table 3. Macro F1 Score Comparison of Bi-LSTM and LLM Feedback Loop methods

Method	Lie F1 Score
Bi-LSTM Approach	24.60
LLM Feedback Loop	26.51

Table 4. Lie F1 Score Comparison of Bi-LSTM and LLM Feedback Loop methods

References

[1] Frank Mendels and Sarah Ita Levitan. Hybrid acoustic-lexical deep learning approach for deception detection. In *Proceedings of the 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 350–356. IEEE, 2017.

[2] Atul Verma, Akash Sharma, Kundan Kumar, R. Dhanalakshmi, and Rajesh Pandey. Deception Detection Using Machine Learning and Deep Learning Techniques. *Journal of Intelligent Systems and Applications*, 3(1):100034, 2024.

[3] Denis Peskov, Benny Cheng, Ahmed Elgohary, Joe Barrow, Cristian Danescu-Niculescu-Mizil, and Jordan Boyd-Graber. It Takes Two to Lie: One to Lie, and One to Listen. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3811–3854, Online, 2020. Association for Computational Linguistics. https://aclanthology.org/2020.acl-main.353/.