Diplomatic Lie Detector

Abstract

In online communication, especially within the strategic environment of the game Diplomacy, accurately detecting deception is a complex yet critical task. Our research extends the foundational work of Peskov et al. by exploring advanced neural network architectures and tokenization strategies to improve the detection of deceit in text-based interactions. Through extensive experimentation, we discovered that a model combining a Bidirectional Long Short-Term Memory (Bi-LSTM) network with a Convolutional Neural Network (CNN), utilizing Unigram tokenization, significantly outperforms other configurations.

This superior performance can be attributed to the synergistic integration of CNN and Bi-LSTM architectures. The CNN component excels in extracting local, salient features within the text, effectively identifying key linguistic patterns that are often indicative of deceptive communication. Concurrently, the Bi-LSTM layer captures the broader context by analyzing the sequence data in both forward and backward directions, essential for understanding the nuanced and often complex nature of deceitful messages.

The integration of Unigram tokenization, distinguishing itself from traditional methods, significantly boosts our model's ability to manage the variability and richness of natural language by breaking text into subword units. This granular approach is essential for capturing the subtle linguistic cues in deceptive communication. Our results reveal that combining CNN and Bi-LSTM layers with Unigram tokenization creates a robust and effective model for text-based deception detection, setting a new benchmark in the field and paving the way for further exploration into linguistic deception detection online.

Introduction

In the dynamic realm of online communication, the detection of deception is both a formidable challenge and a fascinating area of research. The pioneering study by Peskov et al. on unmasking deceit in online interactions, especially in the structured milieu of the game Diplomacy, anchors our investigation. Our study seeks to replicate their groundbreaking work and also broaden its scope by integrating novel analytic techniques and examining a spectrum of tokenization methods in the realm of deep learning.

Central to our inquiry is the exploration of diverse tokenization strategies. Moving beyond the standard Keras tokenizer, we delve into alternative methods such as Byte Pair Encoding (BPE), SpaCy, and the Unigram Language Model Tokenizer. Each of these techniques offers a distinctive approach to text processing, potentially unlocking new insights into the intricate patterns of deceptive language.

In our exploration, we critically analyzed message tokenization in the context of detecting written deception, informed by Peskov et al.'s study which found limitations in BERT's performance. Despite BERT's well-known ability to discern contextual subtleties in language, its effectiveness in unraveling the complex layers of deceptive writing, which often hinge on subtle psychological cues, long-term intentions, and intricate blends of truth and fallacy, remains uncertain. This inadequacy, highlighted in

Peskov et al.'s findings, stems from BERT's training on general datasets, which may not capture the nuanced mechanisms of deceit. Recognizing this gap in understanding the specific markers of deceptive texts, our study ventured into exploring alternative tokenization methods. This strategic decision was driven by the need to address the shortcomings observed in BERT's approach to deception detection and to advance our comprehension of linguistic cues in deceptive communication.

Our research endeavors to make a meaningful contribution to the field of linguistic deception detection. By experimenting with an array of models and scrutinizing different tokenization techniques, we aim to unravel the intricacies of deceit in written communication.

Background

Being able to detect lies and deception is very useful in the real world for things like criminal investigation. Trying to detect if someone is lying is difficult and requires lots of information about the person, their actions, and way they speak. Gupta et al. (2019) developed a multimodal that analyzes video, audio, and gaze to detect if a person is lying. Although their model is very robust, it does not work for something simple like text based messaging to detect a lie. To actively detect if someone is lying in text, a lighter weight and quicker solution would be needed.

Peskov et al. (2020) sought to find a model that would quickly and precisely be able to detect deceit in text based messages. They put together a dataset that was collected from multiple people playing the game Diplomacy and sending messages to each other on Discord. A Discord bot was used as a medium to collect information about the player's message and send the message to the designated recipient. When a player sent a message they were tasked with selecting a thumbs up if they were telling the truth or a thumbs down if they were intending to deceit. The recipient receiving the message would read it and select a thumbs up if they believed the sender was telling the truth or a thumbs down if they believed the sender was lying. The bot also tracked game information with each message for future use.

After collecting labeled text message data, Peskov et al. (2020) developed and tested various model combinations of LSTM with BERT, previous message context, or Power dynamics. They also tested logistic regression and noted how it was interpretable, but it failed to investigate if word sequences can reveal lies. Some of the biggest challenges of detecting a lie in text based conversations is the lack of information. In text conversations there is no tone or emotion and you cannot see the face of the other person. In response to this, things like the context of the previous message were included and tested. Also, some players have more experience or are better at lying than the others. To offset this imbalance, a score differential between the players that were having a conversation is calculated and used as a power dynamic. What Peskov et al. found was that Context LSTM + Power performed the best compared to other models, and it was the only model that performed closest to the human baseline.

Methods

Replicating methods and models Peskov et al. used, we seek to improve on their scores and expand on their work with a variety of methods and tokenizations. The models we implemented from Peskov et al. are LSTM, Context LSTM, and Context LSTM + Power. The models were fit to learn and predict what the sender’s and receiver’s labels were for the message they exchanged. Context from the previous message was added as input to give the model context behind the given message. An imbalance of player experience and skill is evident in the game of Diplomacy. To help offset the skilled players from novice players, a score differential between the two players involved in the interaction was used for Power.

For our first set of experiments, we wanted to test different Recurrent Neural Network architectures for this task. As RNNs are the best architectures for sequential data, we decided to focus on new applications of the LSTM that Peskov et. al used, as well as a Gated Recurrent Unit (GRU). Aside from using two architectures, we also introduced a Bi-Directional Component to the architecture to see if models could learn from future context to make accurate predictions.

In addition to exploring various RNN architectures, we delved into different tokenization methods to process the text data. This included traditional Keras tokenization, as well as more advanced techniques like Byte Pair Encoding (BPE) and Unigram tokenization. These methods break down text into subwords or tokens, offering a balance between capturing the linguistic nuances of full words and the flexibility of character-level representations. We hypothesized that these tokenization techniques might capture different aspects of the text data, potentially leading to improvements in model performance.

Moreover, we utilized SpaCy's linguistic features, specifically Part-of-Speech (POS) tagging, to enrich our models' understanding of the grammatical structure of the messages. By incorporating these POS tags as additional input, we aimed to provide our models with a deeper understanding of the syntactic context of each message.

All models were trained on the training data to maximize the metrics on the validation data, and then subsequently applied to the held-out test set, which is what we reported for performance.

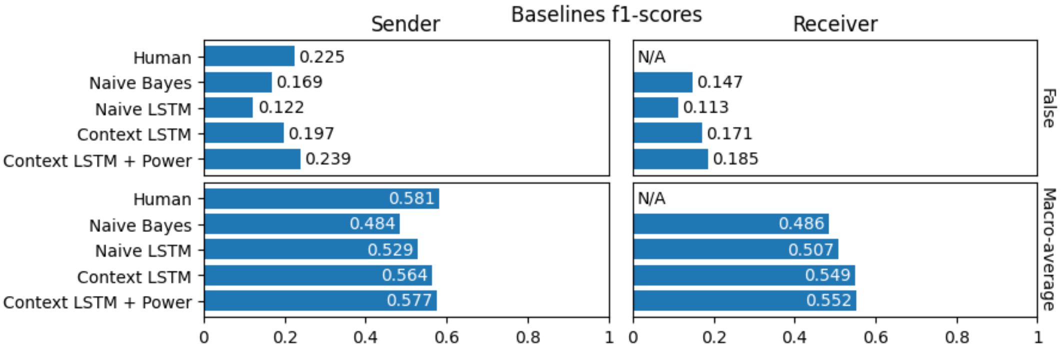
For evaluating our models, we chose two distinct metrics: the macro F1 score and the lie F1 score, each addressing different aspects of our dataset. The macro F1 score is crucial for its equal treatment of all classes, offering a balanced assessment of the model's performance across various labels. This is particularly relevant in our imbalanced dataset, as it calculates and averages the metrics for each label without bias, ensuring that all classes, regardless of their frequency, are represented equally in the evaluation.

In contrast, the lie F1 score is a domain-specific metric, highly pertinent to the game of Diplomacy, where the distinction between truth and deception is pivotal. This score is particularly focused on the model's ability to accurately identify 'lie' messages, a critical component in sender labels. For sender labels, detecting deception accurately is of utmost importance, and the lie F1 score provides a direct measure of the model's efficacy in this aspect.

On the other hand, for receiver labels, where the focus is on understanding whether the message is perceived as truthful or deceptive by the recipient, the macro F1 score's balanced approach offers a comprehensive evaluation. This dual metric approach allows us to tailor our assessment to the nuanced dynamics of sender and receiver perspectives in the game, ensuring a thorough and relevant evaluation of our models' capabilities in both detecting and understanding deception.

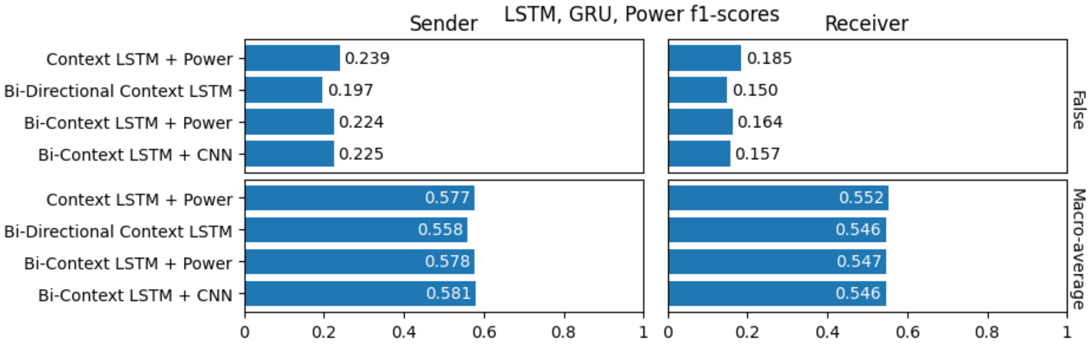
Through these experiments, our goal is to not only improve upon the baseline models established by Peskov et al. and the human baseline, but also to explore how different neural network architectures, tokenization methods, and linguistic features can impact the model's ability to understand and predict the dynamics of player interactions in the game of Diplomacy.

Results and discussion



For our baseline scores, Humans performed the best in predicting if the message was false. This can be attributed to increased context, game awareness, and conversations with other players. Our worst performing model was Naive Bayes, which makes sense since it relies on independence which each message does not have.

Our best baseline model, context LSTM and Power, supports the idea that having context of the previous message is key for determining if the person is lying or not, while also compensating for different experienced players. This mode had a macro F1 score of .577 and a lie F1 score of .239 on the held-out test data. This narrowly trailed our human baseline of .581 for macro F1, but had a higher lie F1. The goal for our final model was to comfortably outpace the human baseline for both macro and lie F1.

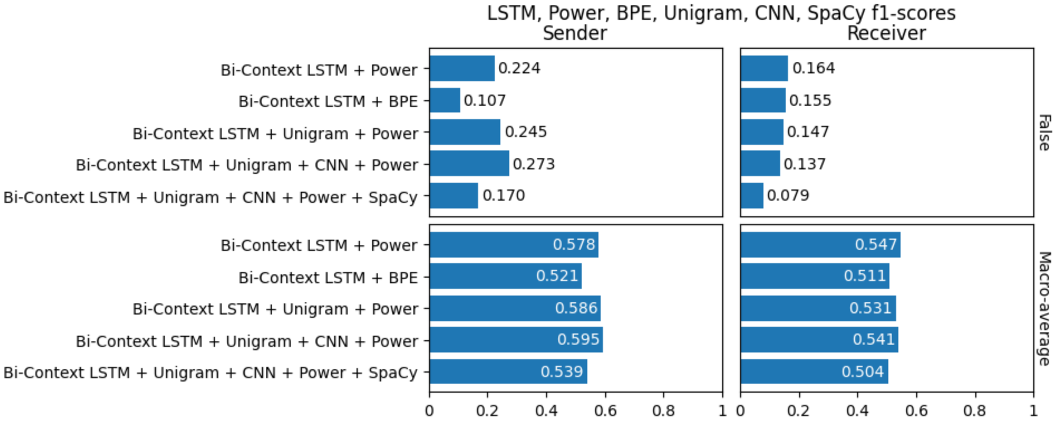


For our first set of experiments, we explored different configurations of Recurrent Neural Networks, specifically introducing a Gated Recurrent Unit, a Bi-Directional component, and a CNN layer beneath the RNN layer as a combination model.

We used a Context LSTM with Power as a baseline for improvements, as it was our best performing model. Both the Context GRU and the bi-directional Context GRU lagged their LSTM counterparts in performance. This isn't too surprising, since GRUs are a bit more simplistic and have 2 gates instead of 3.

When testing a bi-directional component, we found an extremely modest improvement in detecting deception, but a slightly worse performance in detecting whether the receiver believes the message is true. We still decided to proceed with the bi-directional model as we weighed performance on predicting the sender label more than predicting the receiver label.

Lastly, we introduced a CNN layer beneath our bi-directional LSTM, hoping the combination of the strengths of an LSTM and CNN would be uniquely adept at detecting deception. We found that to be the case as it identically matched the human benchmark performance with a macro F1 score of .581 and a lie F1 score of 0.225. It also maintained a similar performance on the receiver label as well.



For our second set of experiments, we wanted to see whether using different tokenization methods other than simple Keras tokenizer. Peskov et. al found that using BERT was detrimental to model performance. Since BERT relies on WordPiece tokenization, we tried other methods such as Byte-Pair Encoding (BPE), Unigram tokenization, and SpaCy tags.

BPE performed very poorly, possibly due to its poor ability to handle rare words. When tested on sender labels, it achieved a macro F1 score of 0.521 and a lie F1 score of 0.107. On the receiver labels, it had a macro F1 score of 0.511 and a lie F1 of 0.155. This barely beat our Naive Bayes baseline overall.

When we incorporated Unigram encoding, however, the model saw a fairly significant improvement, with the macro F1 score on the sender labels rising to 0.586, and the lie F1 score rising to 0.245. This model did perform worse on the receiver labels, however, but we decided to move forward with this model as it outperformed humans in being able to detect lies from both F1 score measurements.

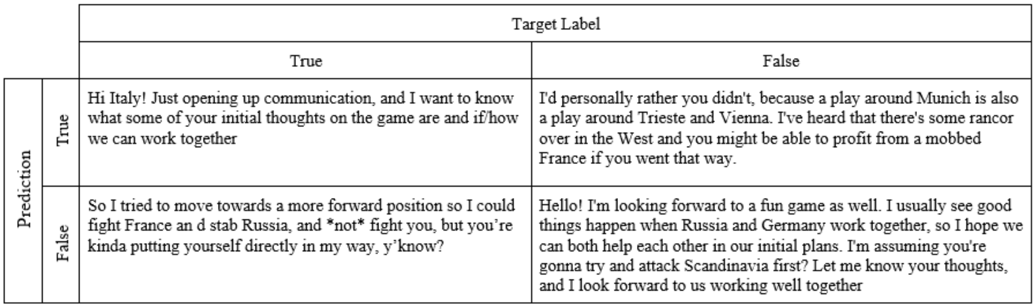
Given we had two standalone models that either met or exceeded our human benchmark, this would be considered a success. However, we decided to incorporate an LSTM into our best-performing Unigram-encoded model to see if the more advanced architecture would perform better. The result was our best performance, with a macro F1 score of 0.595 and and a lie F1 score of 0.273. The macro F1 score for receiver labels was also higher than the general Unigram model, rising to 0.541.

Additionally, we tried to utilize SpaCy tags, namely Part-of-Speech tags, Named Entity Recognition, and Dependency tags, as features. The performance of the model suffered, however, suggesting that it added more noise and was not an adequate method for illuminating the complexity of deceptive language.

In the end, by combining our best architecture with our best encoding scheme, we were able to produce a model that performs comfortably better than a human benchmark.

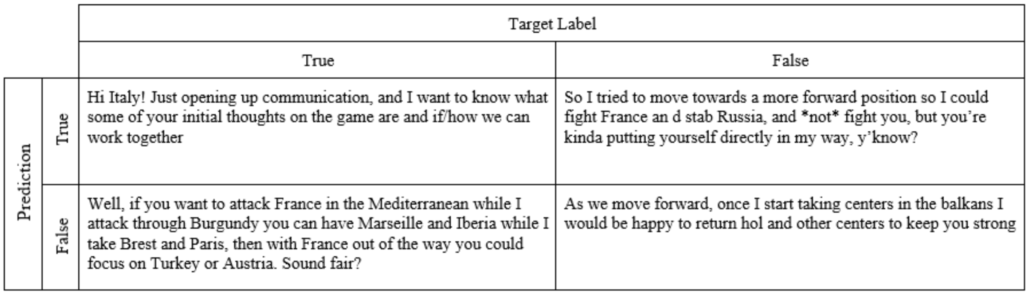
Below are examples of our model's correct/incorrect predictions on both the sender and receiver labels:

Sender Messages



		Target Label	
		True	False
Prediction	True	Hi Italy! Just opening up communication, and I want to know what some of your initial thoughts on the game are and if/how we can work together	I'd personally rather you didn't, because a play around Munich is also a play around Trieste and Vienna. I've heard that there's some rancor over in the West and you might be able to profit from a mobbed France if you went that way.
	False	So I tried to move towards a more forward position so I could fight France an d stab Russia, and *not* fight you, but you’re kinda putting yourself directly in my way, y’know?	Hello! I'm looking forward to a fun game as well. I usually see good things happen when Russia and Germany work together, so I hope we can both help each other in our initial plans. I'm assuming you're gonna try and attack Scandinavia first? Let me know your thoughts, and I look forward to us working well together

Receiver Messages



		Target Label	
		True	False
Prediction	True	Hi Italy! Just opening up communication, and I want to know what some of your initial thoughts on the game are and if/how we can work together	So I tried to move towards a more forward position so I could fight France an d stab Russia, and *not* fight you, but you’re kinda putting yourself directly in my way, y’know?
	False	Well, if you want to attack France in the Mediterranean while I attack through Burgundy you can have Marseille and Iberia while I take Brest and Paris, then with France out of the way you could focus on Turkey or Austria. Sound fair?	As we move forward, once I start taking centers in the balkans I would be happy to return hol and other centers to keep you strong

Conclusion

Our study embarked on a journey to enhance the detection of deception in the game of Diplomacy, a domain where discerning truth from falsehood is paramount. Building upon the foundational work of Peskov et al., we ventured into the realm of advanced neural network architectures and diverse tokenization strategies, aiming to deepen our understanding of deceit in text-based interactions.

The crux of our research led us to a groundbreaking discovery: a model integrating a Bidirectional Long Short-Term Memory (Bi-LSTM) network with a Convolutional Neural Network (CNN) and employing Unigram tokenization emerged as the most effective configuration. This model's superiority lies in its ability to synergize the local feature extraction prowess of CNNs with the contextual and sequential depth provided by the Bi-LSTM architecture. The Unigram tokenization method further augmented this model by adeptly managing the variability of natural language, thereby enhancing its capability to discern subtle deceptive cues woven into text.

Our exploration revealed that while GRUs and other RNN configurations offered insights, they fell short compared to the Bi-LSTM in the context of deception detection. The intricate layers of deceit in written communication, often reliant on long-term semantic dependencies and nuanced linguistic cues, were more accurately captured by the Bi-LSTM framework. Additionally, our experiments with tokenization methods such as Byte Pair Encoding (BPE) and SpaCy tags led to significant revelations about how to approach the task at hand. BPE, in particular, underperformed in this specific task, likely due to its limitations in handling the rarity and complexity of words pivotal in deceptive text.

The culmination of our efforts was a model that not only matched but also surpassed human-level benchmarks in detecting lies, achieving a macro F1 score of 0.595 and a lie F1 score of 0.273. This model stands as a testament to the potential of combining sophisticated neural architectures with advanced tokenization techniques in the nuanced task of lie detection.

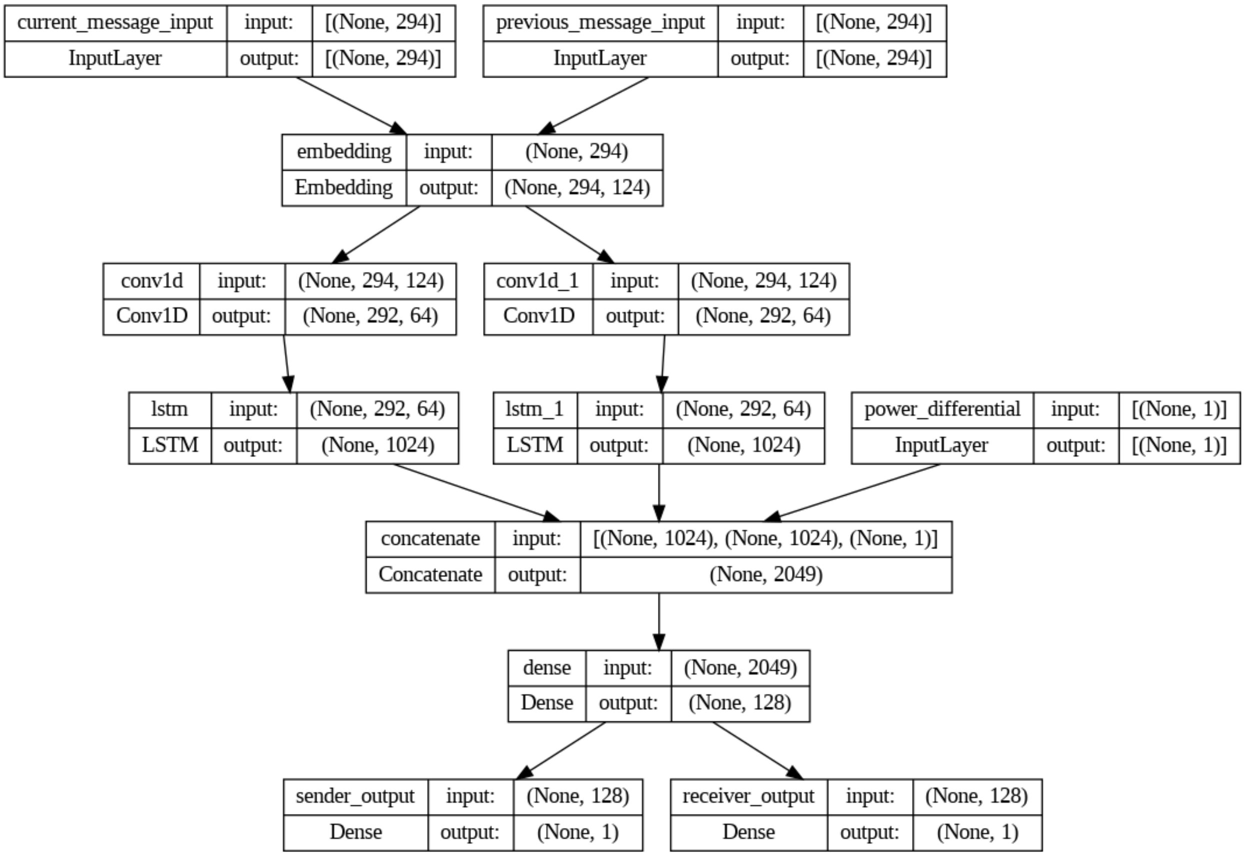
In conclusion, our research contributes significantly to the field of linguistic deception detection. By successfully marrying the strengths of Bi-LSTM, CNN, and Unigram tokenization, we have crafted a model that offers a new benchmark in detecting deception in text.

References

- Denis Peskov, Benny Cheng, Ahmed Elgohary, Joe Barrow, Cristian Danescu-Niculescu-Mizil, and Jordan Boyd-Graber. 2020. 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. Association for Computational Linguistics.
- V. Gupta, M. Agarwal, M. Arora, T. Chakraborty, R. Singh and M. Vatsa, "Bag-of-Lies: A Multimodal Dataset for Deception Detection," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Long Beach, CA, USA, 2019, pp. 83-90, doi: 10.1109/CVPRW.2019.00016.

Appendix

CNN-LSTM Model with Unigram encoding and power differential:


current_message_input | InputLayer | input: [(None, 294)] | output: [(None, 294)]
previous_message_input | InputLayer | input: [(None, 294)] | output: [(None, 294)]
embedding | Embedding | input: (None, 294) | output: (None, 294, 124)
conv1d | Conv1D | input: (None, 294, 124) | output: (None, 292, 64)
conv1d_1 | Conv1D | input: (None, 294, 124) | output: (None, 292, 64)
lstm | LSTM | input: (None, 292, 64) | output: (None, 1024)
lstm_1 | LSTM | input: (None, 292, 64) | output: (None, 1024)
power_differential | InputLayer | input: [(None, 1)] | output: [(None, 1)]
concatenate | Concatenate | input: [(None, 1024), (None, 1024), (None, 1)] | output: (None, 2049)
dense | Dense | input: (None, 2049) | output: (None, 128)
sender_output | Dense | input: (None, 128) | output: (None, 1)
receiver_output | Dense | input: (None, 128) | output: (None, 1)