December 21, 2020 Advised by Denis Peskov

Fall Semester Research
A Working HITL Diplomacy Bot

Sander Schulhoff
University of Maryland

1 Creating a Human-in-the-Loop Diplomacy Bot

As an extension to past Diplomacy work (Niculae et al., 2015; Peskov et al., 2020; Paquette et al., 2019; Anthony et al., 2020), we want to add a human-in-the-loop component that provides players with insight from a lie detection model. The motivation for this is to help a user *in the moment* better decide whether they are being deceived We needed to change both the functionality (Section 2) and the display (Section 3) of the Discord bot. After making these changes, we conducted an Alpha test. In order to complete these tasks, I had to develop a number of additional skills as outlined in Section 7.

2 Functionality Changes

When I began working on the bot, there were a few functionalities that needed to be debugged or implemented. First, the function to get data from the server, 'fetch', was not working. Importing the correct 'node-fetch' function resolved this. With data being correctly fetched, the only problem was returning it to a function which would embed the information in the bot's message. Since fetch calls are asynchronous, this was a bit difficult. I asynchronized the function in which fetch was called (the .once call of a Discord.js reaction collector) and used the 'wait' keyword to force the program to wait until data was returned to continue processing instructions. This resolved all functionality issues. I also added error handling so that if there are any errors with the Python server or if it is down, the Discord bot will just display the original message type as seen in Figure 1. Additionally, I changed the database service to a MongoDB Atlas cluster as the old cloud host stopped functioning.

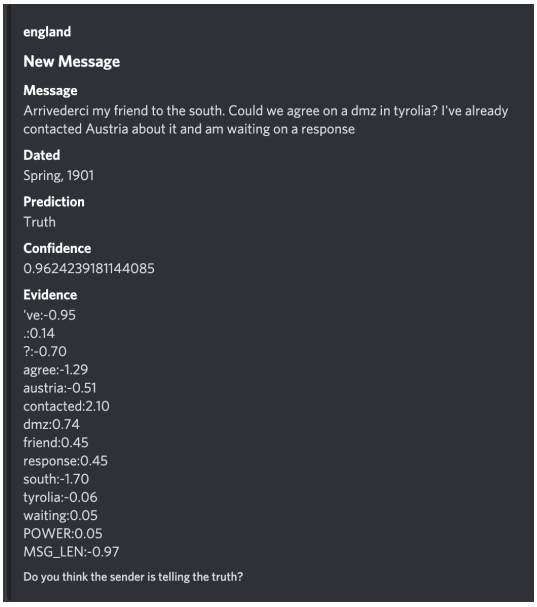
3 Display Changes

This section describes our design choices, with the human-in-the-loop interface inspired by (Wallace et al., 2019) and design principles inspired by (Smith-Renner et al., 2020; Sundar, 2007).

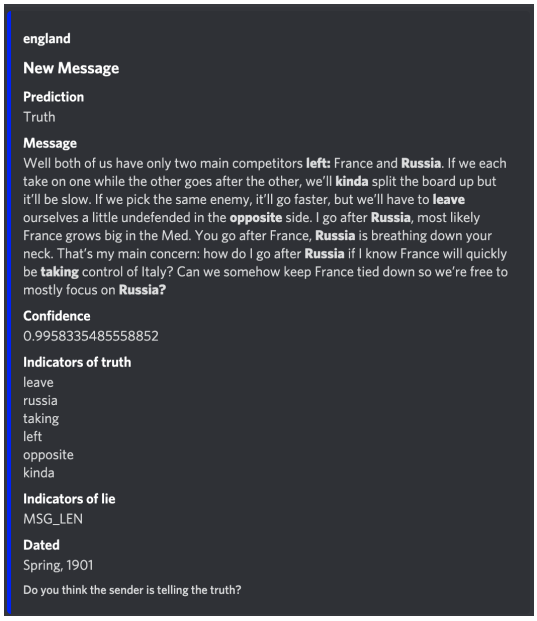
Order The format and design of the bot message were the next items that needed to be changed. Originally, the message was displayed with unmarked-up words and unordered evidence containing words and their values (wordscores). I ordered the evidence by its weight and displayed the top five evidence words that support the bot prediction. This formatting was changed later with more comprehensive display functionality which is described below.

Coloring The color of the embed is important to help a user quickly distinguish whether a message is a truth or a lie. I added embed coloring based on bot prediction (green for true, red for false). I later changed the coloring to blue for true and red for false to deal with possible color-blindness (Wong, 2011).

Evidence Evidence words are marked up in messages.



(a) Example of original message. Evidence is not marked up in the actual message and the evidence words are uncapped and unordered.



(b) Example of the final message version (a truth)

Figure 1: Original and most recent versions of a message

Cleaning It is necessary to clean evidence wordscores returned by the model as not all of them are helpful for a user. Evidence wordscores returned by the model which are only punctuation or have a score of 0 are removed. Parts of contractions returned as evidence are dealt with by marking up the whole contraction.

Truth vs. Lie Truth words and Lie words are marked up differently in messages. If a word is indicative of truth, it is bolded. If it is indicative of a lie, it is underlined. The amount of truth and lie words marked up is dependent upon the length of the message sent and the strength of the prediction. Not all of the evidence words returned by the Python server (after cleaning) are marked up.

4 Maximal Evidence Cap

To ensure the visual appeal of messages and that users do not get overloaded with information (Kortschot et al., 2020), it was necessary to cap the amount of evidence words marked up.

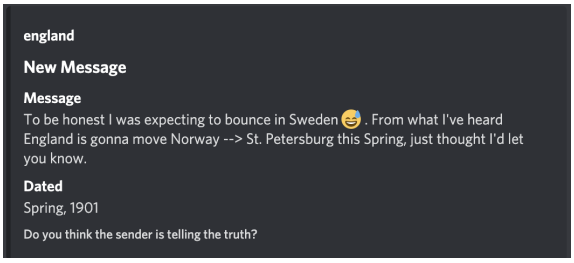
Calculations The total amount of distinct evidence words marked up (TDEW) in a given message is approximately equivalent to the natural logarithm of the length of the message. If the model predicts truth, the number of distinct truth words marked up in the message will be approximately half the TDEW plus the product of half of the TDEW and the model confidence (which is on the interval [0, 1]). The amount of distinct lie words to be marked up will then be calculated by subtracting the truth words to be marked up from the TDEW.

Reasoning Originally, there had been a value threshhold so that evidence words of little significance wouldn't be marked up. However, I removed this as it allowed for the prediction to display as truth, but only show evidence indicative of a lie. Also, the logarithm controls the amount of words marked up perfectly well without the threshhold. I use the natural logarithm to compute the TDEW because as the amount of distinct words which are marked up grows, the total amount of marked up words grows even faster (According to my calculations, logistically faster). This is due to the repetition of words in the message and the fact that the current

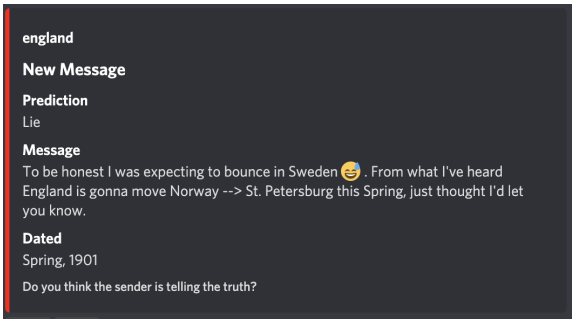
production model does not distinguish a particular instance of its selected words to be the one indicative of a truth or lie.

Results The complexity of this system creates visually appealing marked up messages where the amount of evidence marked up is not overwhelming and the ratio of lie evidence to truth evidence is apparent (and corresponds to the prediction confidence), especially in longer messages. I tested a number of messages from the data set and found that in practice this system does generate visually appealing messages.

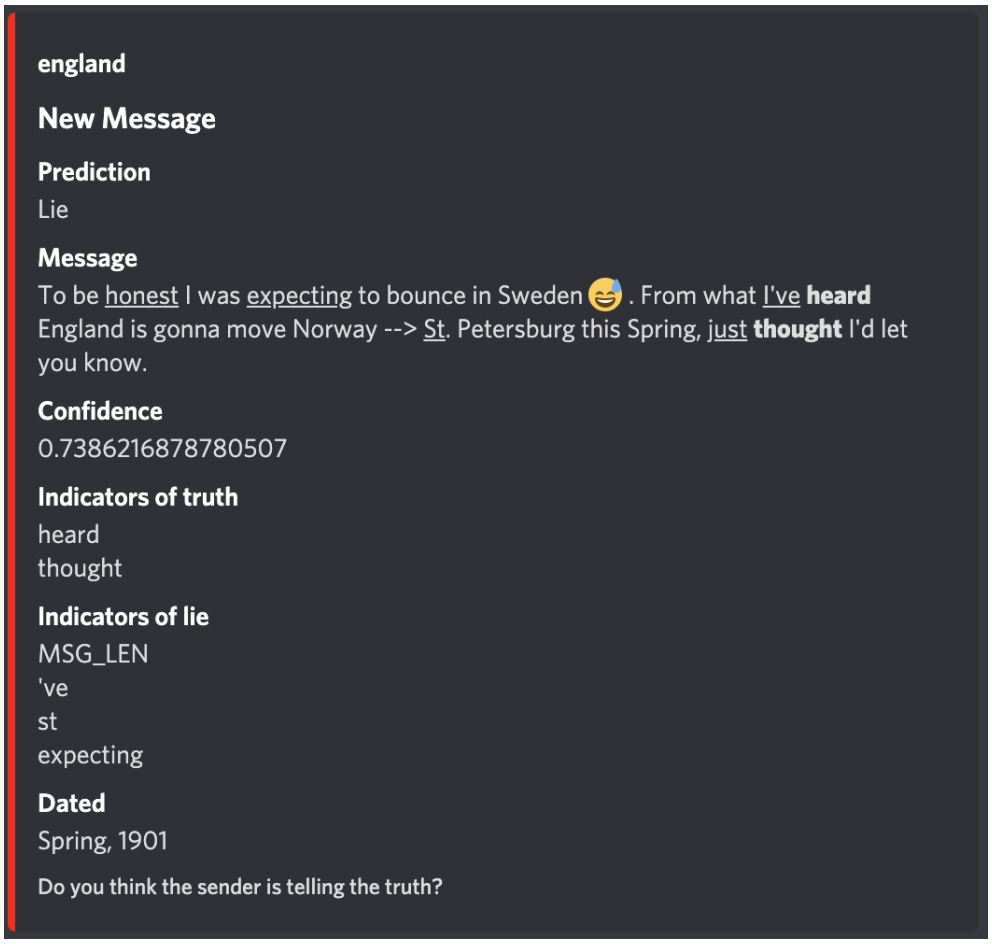
5 Message Display Types



(a) Message only



(b) Message with prediction



(c) All information

Figure 2: Different versions of a lie message

For the purpose of conducting studies on the potential player benefits of seeing bot predictions, I made a few different message display types which each display different information. Every time the bot sends a message, there is an equal chance of each message type being sent. The different types will be used to test the statistical significance of the different interfaces.

Message Only This message type contains no prediction or evidence information. The message is displayed, but no words are marked up. An example is shown in Figure 2a.

Message with Prediction This message type contains the message, the prediction, and the colored embed. Lies are highlighted with red and truthful messages are highlighted with blue. However, no evidence is displayed, nor are words marked up. An example is shown in Figure 2b.

All Information This message type contains all information, including prediction words, markup, and confidence. An example is shown in Figure 2.

6 Alpha Test

In order to test the functionality of the bot, Denis and I organized an Alpha test with 6 players with experience ranging from entry level to veteran. We collected dozens of messages from the players with varied characters like emojis, Chinese characters, and other Unicode characters. We tested the different message types (message only, message+prediction, all information). So far, the only bugs have been characters such as "=" and "\n" appearing as evidence and the markup messing up on one message, seemingly as a result of the former bugs. I have begun collecting input from the users. One suggestion that emerged was to allow the bot to show "unsure" as the prediction label, especially in cases where the message is very short.

7 Building a NLP Toolkit

Entering college, I was not very familiar with Natural Language Processing and had never played Diplomacy. To begin this project and ultimately become a researcher in the field, I needed to build my skill-set. Towards this, I developed the following skills:

Learning Diplomacy I read the Diplomacy rule book so I could be familiar with the subject area of my research. I then played a game on Backstabbr, which is the platform for our study. This taught me about lingo in Diplomacy and player communications.

Maryland Infrastructure Since the bot is hosted on Maryland servers, I had to set up a UMIACS account and read through the docs on how to login and run the Discord bot. For the future semester, my goal is to learn more about the cluster for heavy computation.

Web Development Backend database storage is an integral part of the Diplomacy project. To develop my familiarity with backend infrastructure, I helped implement a working website for Denis' modulation project using the Wikipedia autocomplete API, HTML, CSS, Javascript, and PHP: wikipedia-autocomplete-site. While the user is typing answers into a text input, autocomplete suggestions are being displayed in a menu that drops down from the text input. It is published here: http://modulation.rf.gd. This experience taught me how to write clear instructions for users and how to test the accuracy of a database in an Alpha test.

Hackathon I participated in the UMBC hackathon and won the Cipher-Tech prize: https://devpost.com/software/foretrackr. I did the frontend for this project and the realization that I was pretty unskilled inspired me to start learning more professional webdev (node, bootstrap, react, etc.) and I started this specialization course: https://www.coursera.org/learn/front-end-react. I plan to finish it by the end of the break.

Dither Videogame For my Digital Studies class, a classmate and I made the game Dither: https://flyingpengun.itch.io/dither which involves levels of logic puzzles where you push around 1s and 0s to satisfy logic gates. As a piece of art, the game is supposed to address the limitations

in binary thinking for question answering. I implemented fetch calls to a trivia API in order to generate infinite levels (level generation is dependent upon a natural language question associated with the level).

GAS program I have been working to bring to market a 1000-2000 line suite of Google Apps Scripts I wrote from scratch for my high-school last year to assist with the process of matching students to their teacher recommenders for college. I converted my code into a Google Sheet Add-on with a MaterializeCSS-styled sidebar. This project should be finished by the end of the upcoming break.

Summer Preparation To continue developing as a computer scientist, I am looking for an internship for this summer. I do not have real working experience, so Denis and I practiced interviewing and he helped improve my resume. I am currently interviewing with WillowTree and am considering Johns Hopkins University for a summer position. Both should allow me to continue building my technical skill-set if I am hired.

8 Summary

Here is a concise list of the skills/frameworks/programs I used or learned this semester: Python, Javascript, PHP, HTML, CSS, Bootstrap, MaterializeCSS, Node, Discord.js, Flask, PyGame, GAS, Postman and fetching data with APIs, Figma, AdobeXD, MongoDB, UMIACS, LaTeX, VSCode, Interviewing, CMSC132 course subjects.

References

Thomas Anthony, Tom Eccles, Andrea Tacchetti, János Kramár, Ian Gemp, Thomas C. Hudson, Nicolas Porcel, Marc Lanctot, Julien Pérolat, Richard Everett, Roman Werpachowski, Satinder Singh, Thore Graepel, and Yoram Bachrach. 2020. Learning to play no-press diplomacy with best response policy iteration.

Sean W. Kortschot, Greg A. Jamieson, and Amrit Prasad. 2020. Detecting and responding to information overload with an adaptive user interface. *Human Factors*, 0(0):0018720820964343. PMID: 33054359.

Vlad Niculae, Srijan Kumar, Jordan Boyd-Graber, and Cristian Danescu-Niculescu-Mizil. 2015. Linguistic harbingers of betrayal: A case study on an online strategy game. In *Association for Computational Linguistics*.

Philip Paquette, Yuchen Lu, Seton Steven Bocco, Max Smith, Satya O-G, Jonathan K Kummerfeld, Joelle Pineau, Satinder Singh, and Aaron C Courville. 2019. No-press diplomacy: Modeling multi-agent gameplay. *Advances in Neural Information Processing Systems*, 32:4474–4485.

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 *Association for Computational Linguistics*.

Alison Smith-Renner, Ron Fan, Melissa Birchfield, Tongshuang Wu, Jordan Boyd-Graber, Daniel S Weld, and Leah Findlater. 2020. No explainability without accountability: An empirical study of explanations and feedback in interactive ml. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

S. Sundar. 2007. The main model : A heuristic approach to understanding technology effects on credibility.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019. Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering. *Transactions of the Association for Computational Linguistics*, 7:387–401.

Bang Wong. 2011. Color blindness. *nature methods*, 8(6):441–442.