May 13th, 2020 Advised by Denis Peskov

Spring Semester Research
An Improved HITL Diplomacy Bot

Sander Schulhoff
University of Maryland

1 Improving The Human-in-the-Loop Diplomacy Bot

A human-in-the-loop interface will supplement past Diplomacy work (Niculae et al., 2015; Peskov et al., 2020; Paquette et al., 2019; Anthony et al., 2020). These changes, inspired by user experiences in the Alpha test, discussion with my advisor, and feedback from my PI, should be the final changes to this iteration of the Diplomacy Bot, unless the Beta test reveals the need for additional changes. We changed aspects of both the functionality (Section 2) and the display (Section 3) of the Discord bot. We will begin a Beta test shortly upon final IRB approval.

In addition to my primary responsibilities regarding the Diplomacy project, I developed an annotation workflow for the QANTA team, published an app, and learned a variety of new skills as detailed in Section 9.

2 Functionality Changes

Although the Diplomacy Bot was technically functional, it was apparent from the Alpha test that the user experience could be improved.

Regex The code the Diplomacy Bot used to mark up text indicative of a truth/lie parsed the message by spaces in order to locate and markup individual words. This was an inefficient and fragile method. For example, with the current Python prediction model, the word "isn't" is tokenized into two words ("is" and "n't"). Either of these words may be targeted for markup, and the markup script needs to account for this. By implementing Regex replacements, this type of markup is now possible. Adding Regex also fixed a number of odd-looking markup behaviours.

Power Deltas Power Deltas (AKA Power Differences or Supply Center Deltas) are a feature included in the training of the model. A Power Delta is the integer difference in Supply Centers between two countries during a communication (a message sent or received). In the original Bot from the Diplomacy project (Peskov et al., 2020), Power Deltas were scraped at game completion. Thus, there did not exist a way of scraping the appropriate information and computing the Deltas in real time (during gameplay). The model training script used Python and Beautiful Soup to perform the scraping/computations. Since the Discord Bot is written in Javascript, I needed to rewrite these scripts in Javascript for the purposes of efficiency and ease of future modifications to the Bot. I started learning JSSoup (a BeautifulSoup copy), but switched to scraping the raw HTML from Backstabbr and running simple Regex queries to extract the Supply Center information. I save scraped information into a global dictionary so that power differentials can be quickly computed when a message is sent.

Data Saving I updated the Message Schema to include the timestamp to assist with game state reconstruction (Zhang et al., 2021) in order to be consistent with work on no-press Diplomacy (Paquette et al., 2019). This improves the ability to reconstruct the exact state of the game (in terms of Messages being sent) and ensures that the order in which messages are sent is

being saved accurately in case there are any discrepancies concerning when messages are saved into the MongoDB Atlas database due to possible unpredictable latency.

Additionally, now that the Diplomacy Bot is scraping Power Deltas each round, I am saving the Deltas in the database as a new Schema type. This will help future state reconstruction and model training as all map data is now being collected.

Python Server The Python prediction server previously returned a JSON object containing three fields: the prediction, the confidence, and all of the evidence. The server sent two special evidence words, MSG_LEN and POWER, in the same JSON entry where evidence words from the actual message were sent. This could create an issue if a user used the word POWER in their message as the Diplomacy Bot would be unable to distinguish whether the model felt the word POWER was important or it felt that the power differential was important. It also made evidence processing more difficult.

I added an additional entry to the JSON called additional_evidence and made the server always put MSG_LEN and POWER there if it decided that they are evidence. I also wrote new Bot-side code to deal with this. This required identifying these special features in the process of selecting what evidence to display (based on message length and weights), while making sure that they were treated separately from regular evidence words. To accomplish this, every time I iterated through the loop which selects evidence to markup, I checked if the weight(s) of the additional evidence was more significant than the current word the loop was checking. If so, I include the special evidence word, decrement the amount of words left to markup, and continue the loop from the same place.

Code Base I cleaned up a number of code snippets, including providing appropriate Javascript Block Scope keywords to a number of variable declarations, improving efficiency of code, and adding detailed explanatory comments.

3 Display Changes

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

Uncertain Prediction The ability for the Discord Bot to display its Prediction as Uncertain was a topic discussed by some of the players during the Alpha test and was later discussed in Pinafore group meetings. After consulting with Denis and reading works on uncertain predictions like (Krstajic et al., 2017), I added this feature so that when the confidence provided by the model does not exceed .70, the Bot displays its Prediction as Uncertain and does not display the confidence; displaying confidence without a predicted label would be confusing to the user. Evidence indicators are still displayed in this case (Figure 1).

Additional Evidence The Python prediction server provides two features which are not words from the message as evidence. These two features are POWER and MSG_LEN. When they are provided to the Diplomacy Bot as evidence, they signify (respectively) that the Bot feels that the difference in supply centers is important or that the length of the message is important.

Previously, these features were displayed in the same field (Indicators of Lie/Truth) as other evidence words. They would appear with only their name and their weight. Now, when they are displayed, they are each displayed as separate fields, which more directly use the actual values of MSG_LEN and POWER to explain their significance to the user.

The MSG_LEN field is now called the Message Length field and uses the average message length (computed across the train data) to display a message in the following format: The message length is {X}% of the average message length. This difference is indicative of a {Truth/Lie}, with a weight of {weight}.

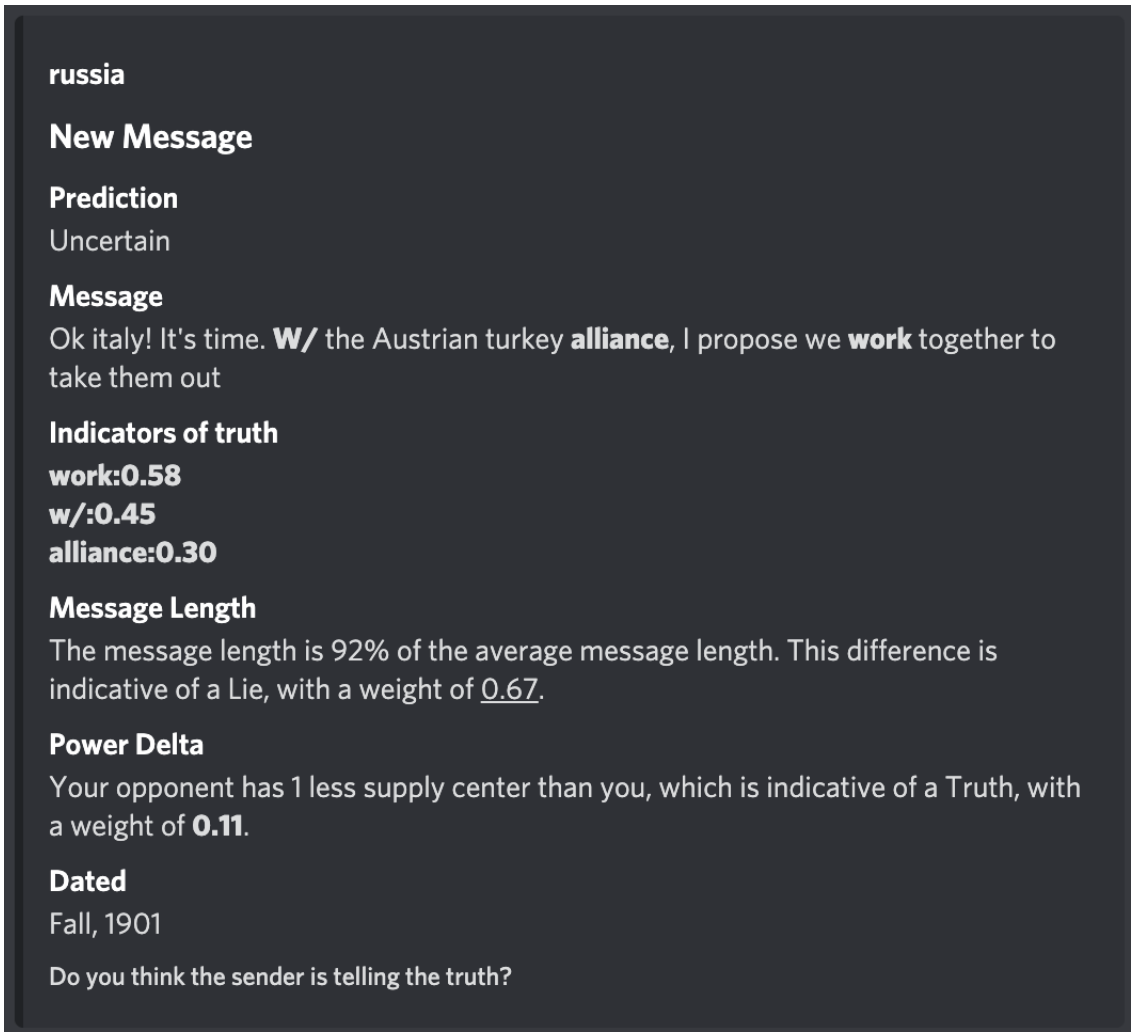

Figure 1: Image of a marked up Diplomacy Bot message containing an "Uncertain" prediction as well as the Power Delta and Message Length fields.

The POWER field is now the Power Delta field and displays a message in the following format: Your opponent has {X} {more/less} supply center{s} than you, which is indicative of a {Truth/Lie}, with a weight of {weight}.

Figure 1 contains an example of a message which contains both of these fields.

4 Diplomacy Bot Instructions

To explain the new features in messages created by the Discord Bot (prediction, evidence, etc.), I wrote an additional Discord embed which is sent just after the general use instructions at the start of a game (Figure 2).

5 Diplomacy Bot Scripts

This semester, I wrote and maintained a number of scripts to analyze and display Diplomacy data.

Confusion Matrix Script I wrote this script to iterate through all of the annotated Diplomacy message data, run predictions through the model, and categorize the resultant data into a quadrant of the confusion matrix table. This was useful for evaluating the model's prediction

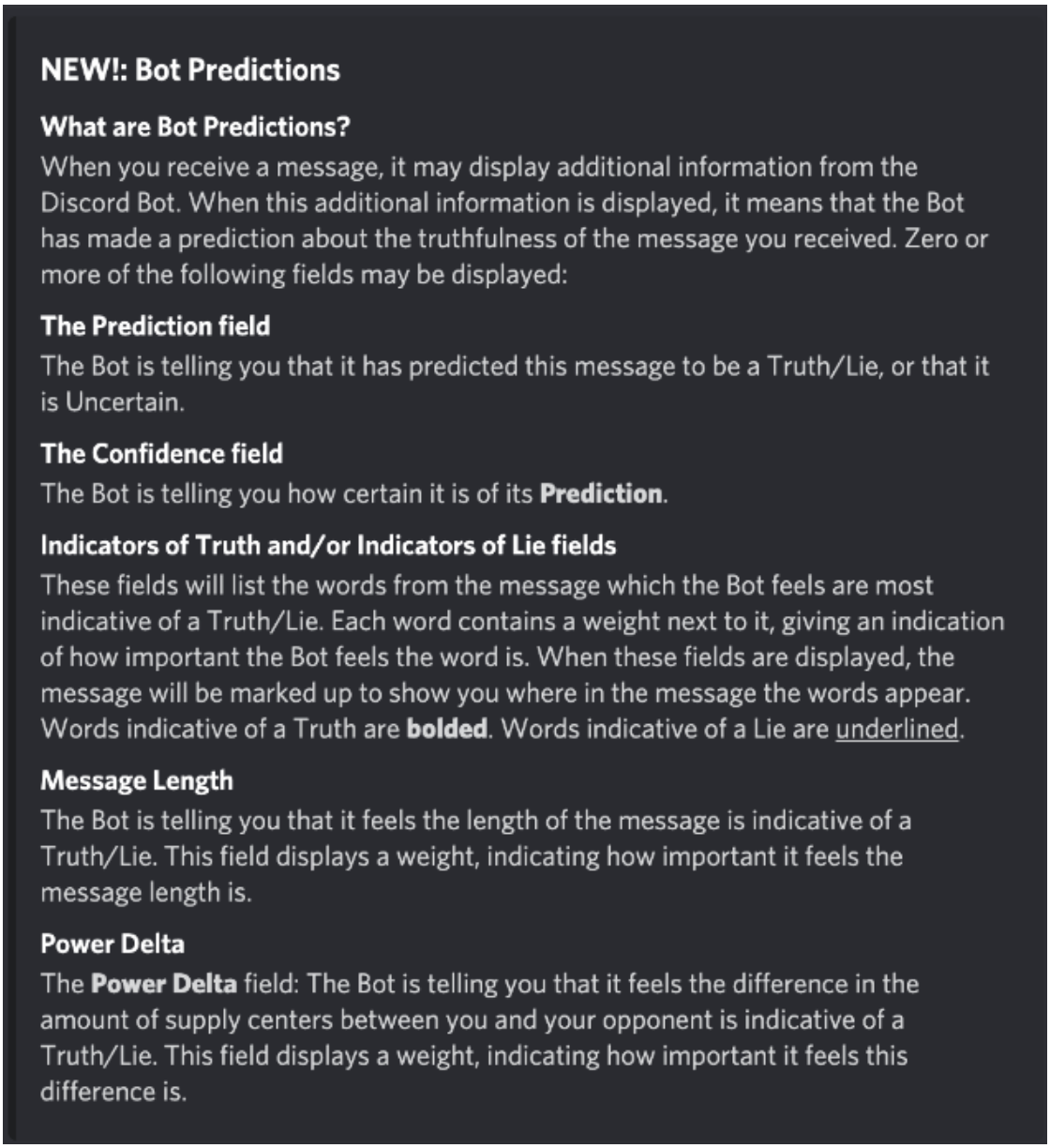

Figure 2: Prediction Instructional Message sent in Discord server

success and ensuring it was functioning correctly.

Average Message Length Script This is a simple script which, initially, read all the annotated Diplomacy messages and found the average of their length. At the PI's request, I changed it to only look through the data used to train the model. The average message length stayed the same. The Diplomacy Bot uses the average message length when the Message Length field is displayed, in order to describe how the length of the message the user received differs from the norm.

PDF Generator Script I made maintenance changes to this script most often because it needed to reflect any changes to the message display in the most recent version of the Diplomacy Bot. This script reads through all annotated Diplomacy messages, passes the messages into the Lie detection model, and displays each message as similarly as possible to how it is actually displayed in the Discord server (i.e. the same fields and text are displayed, but the coloring and spacing may differ). The messages are organized according to their confusion matrix quadrant, marked up as they would be in Discord, then converted by a Discord formatting converter library into HTML. This HTML file can then be exported as a pdf.

6 Code Documentation

As part of my work with the Diplomacy Bot, I edited the README.md instructions and created additional instructions so that future users of this code base can easily run the Bot and Server. I wrote one set of instructions specifying the exact commands to run in order to run the Bot and the Server on UMIACS. Additionally, I rewrote mongoexport scripts from scratch so future researchers can easily download the data from the MongoDB Atlas database.

7 Model

To continue the development of the prediction model for the Discord Bot, I experimented with concatenating past messages to provide context (Garten et al., 2019).

X past messages concatenation For each message in the train data, I appended to it the last X messages sent in the chat. I wanted to see if this conversational context would be sufficient to improve prediction ability. Ultimately, it was not. For most values of X, precision and recall for truths and lies went down. For some values of X, they increased slightly, but I was not able to improve lie prediction, which was my main goal.

X past messages from enemy concatenation For each message in the train data, I appended the X last messages sent in the chat by the same person who sent that message. I hoped that Bot being able to see multiple messages from one player would improve its prediction. However, I saw similar results to my past strategy– macro average decreased for all values of X which I tried.

Modeling Trust in Diplomacy: Reflections from a CSS perspective I read through Xin Qian's paper and considered the features of hedging described by (Stiles, 2018), empathy using analysis from (Sedoc et al., 2020), and emotional intensity which uses an emotional intensity lexicon from (Mohammad, 2017). I found these features to be very interesting and seemingly very useful. Including these features in the next iteration of the Diplomacy Bot would take us significantly closer to a Bot which clearly explains its decisions.

To this end, I have adapted Xin's methodology and begun training the model using her additional features. I created a new branch for the Diplomacy Bot and implemented this new model as well as Bot display changes to take into account the additional features (Figure 3).

I performed some analyses of the use of hedging words (words from (Stiles, 2018)'s lexicon) and found that 63% of actual Truth messages contained hedging words and about 75% of Lie messages contained hedging words. I compared the models with hedging and without hedging according to their confusion matrices and found that the two models performed comparably.

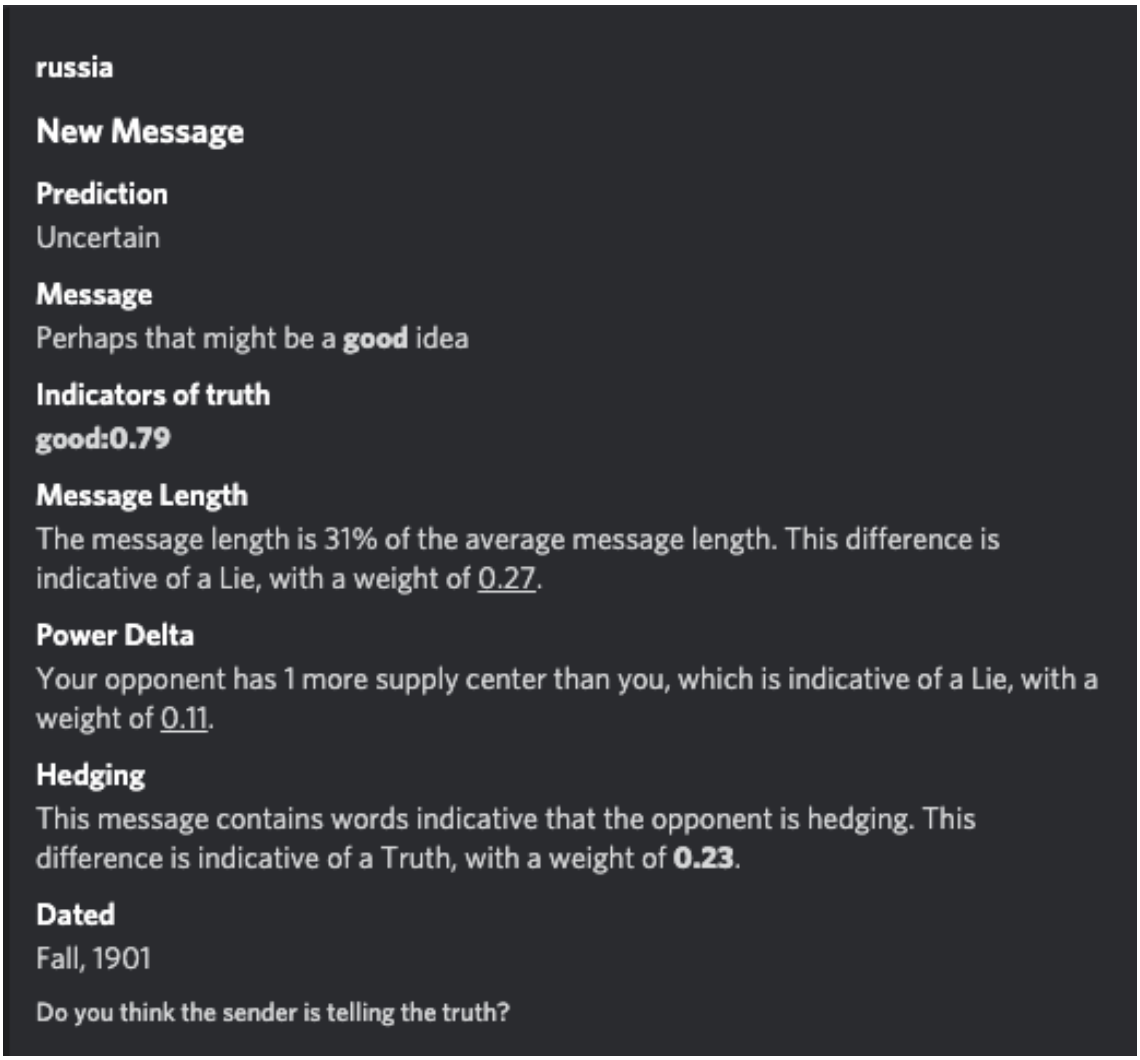

Figure 3: A message sent by the Discord bot which includes the hedging field.

I look forward to seeing all of the additional features fully implemented. Even if they don't significantly improve prediction ability, there is value in improved explanation to the user.

Future of the Model Additional features and algorithms can improve the model. Graph Neural Networks (Zhou et al., 2021), as used in (Anthony et al., 2020), would provide context on a higher level (specifically that of the entire game board) than the current POWER feature, which is just the integer value representing the imbalance in supply centers between two players. A temporal-spacial feature, which predicts correlations between a player's language over time and their movements on the game board, could also be very useful. This might manifest as an additional model which actually predicts movements based on past movements and messages. Although I likely don't have the technical acumen to effectively implement these algorithms currently, they are certainly things I will think about as I improve my ML proficiency.

8 Form Revisions

As part of my semester work, I took part in reviews of a number of academic forms. I read over "Proposal—Technical and Management Volume 1 Cover Sheet", noting possible errors. I went through the Diplomacy IRB after the committee most recently noted needed changes. I made the necessary changes. I also read/edited an unrelated proposal by Ryan Cotterell about

Gendered Language.

9 What I Learned

From completing the Diplomacy Bot to publishing a Google Application, my developer skill-set has significantly expanded. Here are a number of skills and tools I learned to use this semester.

Diplomacy Over this past semester and past year, I have significantly improved my capability to work with asynchronicity in both JavaScript and Python. I have also learned a lot about Discord.js. I have improved my communication and presentational skills by generating images or scripts to produce text data and presenting them in meetings and reports. I have seen the complexities of the research process and how many different steps (like the IRB) one must go through before even beginning an experiment.

QANTA annotation workflow As part of a team of undergrads working on Pedro's QANTA project, I worked to build a data annotation workflow so the QANTA team can easily collect QANTA data from users. I used Label Studio to implement a workflow where the user is presented with a Jeopardy question and its answer. Then, the user is asked to put the corresponding Wikipedia page into a text box. Though it is a very simple workflow, I discovered a bug with the latest version of Label Studio (which they had just published). The question and answer fields would both display as either the question or the answer. After consulting with the Label Studio team, they quickly solved the issue and pushed a fix. Through this experience, I learned how to quickly learn and apply a new tool as well as how to present a minimal reproducible example and explain my code problems.

Latex Notes to Anki This year I frequently changed my note taking style as I was trying to find what works best for me. As part of this search, I started taking notes in LaTeX. I decided that notecards would be useful as a study tool, so I wrote a Python script which parses my LaTeX notes using pylatexenc then converts them into flashcards. I created a few Macros to structure my notes. When I take notes using these Macros, the script can convert them into two basic types of cards (depending on which Macros I am using). One Macro, word{}{}, takes a word then its definition. If I pass strings to both, the Python script converts this to a simple flash card containing the word on the front and the definition on the back. If I include multiple onefact{} Macros within the definition section, the Python script converts this information into a card which on one side says "List X pieces of information about TOPIC". The other side contains an enumerated list of the facts included in the onefact{} Macros. Additionally, the script includes path information on the fronts of the note cards (i.e. what section and subsection of the notes is this definition in). The note cards can also contain LaTeX expressions. This project was somewhat difficult as the parsing library does not create a complete tree structure, but rather each node in the tree it creates has a list of subnodes. It was somewhat confusing to keep track of path information as \section and \subsection don't have children so I needed to write extra code to keep track of the current section and subsection as my script iterated through the notes. Completing this project helped me learn about Python, Latex, and how I learn.

10 Summer Work

I have been hired to work on applications of situated intelligence (with the Microsoft Psi framework). I will be working with a group of grad students under Professor Andreas Andreou at Johns Hopkins University. I will also be helping Denis with any data collection activities and debugging as is necessary.

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.

Nicholas Botzer, Shawn Gu, and Tim Weninger. 2021. Analysis of moral judgement on reddit, 01.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable AI for natural language processing. *CoRR*, abs/2010.00711.

Maria Dymitruk. 2019. Ethical artificial intelligence in judiciary. 02.

Justin Garten, Brendan Kennedy, Kenji Sagae, and Morteza Dehghani. 2019. Measuring the importance of context when modeling language comprehension. *Behavior Research Methods*, 51, 02.

Damjan Krstajic, Ljubomir Buturovic, Simon Thomas, and David E Leahy. 2017. Binary classification models with "uncertain" predictions.

Saif M. Mohammad. 2017. Word affect intensities. *CoRR*, abs/1704.08798.

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, Steven Bocco, Max O. Smith, Satya Ortiz-Gagne, Jonathan K. Kummerfeld, Satinder Singh, Joelle Pineau, and Aaron Courville. 2019. No press diplomacy: Modeling multi-agent gameplay.

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*.

João Sedoc, Sven Buechel, Yehonathan Nachmany, Anneke Buffone, and Lyle Ungar. 2020. Learning word ratings for empathy and distress from document-level user responses. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1664–1673, Marseille, France, May. European Language Resources Association.

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.

Kendall Stiles. 2018. *Trust and Hedging in International Relations*. University of Michigan Press.

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

Anna Tigunova, Andrew Yates, Paramita Mirza, and Gerhard Weikum. 2019. Listening between the lines: Learning personal attributes from conversations. pages 1818–1828, 05.

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.

Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. 2021. Robust reinforcement learning on state observations with learned optimal adversary.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2021. Graph neural networks: A review of methods and applications.

11 Appendix

Two additional projects helped me develop skills useful for Natural Language Research.

GAS program As part of the UMD Venture Practicum course, I completed development of my Teacher Recommender System (TRS) application. After a long Google review process, I finally got it published on the Google Workspace Marketplace here. I had to go through sensitive scope review processes and make a number of updates to the TRS website and codebase in order to satisfy their criteria. My final codebase contains ~4500 lines of back-end Google Apps Script and front-end Javascript/HTML/MaterializeCSS. I will be deploying the app in at least one highschool this year. I learned a lot from the completion of this project, including the coding languages involved, business practices, law practices regarding data privacy and terms of use, how OAuth scopes work, and how to get through an official verification process. I also created a Limited Liability Company to support the business I am creating with this app. The use of apps is relevant to current research efforts such as KARL.

Mini NLP project One Reddit subreddit which I used to browse, AITA (Am I The Asshole?), seems to be a goldmine for NLP model explainability research (Danilevsky et al., 2020). Users on the subreddit post a story involving themselves where they took some action which was controversial to the people around them. Internet users comment to post a judgement, j $\in$ {NTA = "Not The Asshole", YTA = "The Asshole", ESH = "Everything Sucks Here" AKA everyone was an Asshole.}. Then, they post an explanation of why they made this judgement. There are millions of posts, often with thousands of comments on them. After the community comes to an agreement, moderators label the post according to the previously mentioned labels.

I scraped ~1000 'Hot' (currently very popular and commented on) posts using the PRAW scraping library. PRAW let me easily scrape the title, label, message body, and comments for each post. Unfortunately, this library/Reddit has a daily scrape limit.

Once I collected this data, I removed posts which didn't have a label, or were labeled ESH, and trained a simple Bag of Words binary classifier. It was able to get ~70% accuracy with minimal pre-processing. I think that building a better classifier and then training GP2/3 or another text generation model (to generate judgements on stories like those in the posts) on this data could produce very interesting results, as well as 1-2 papers. I look forward to maturing mathematically to be able to write these papers.

A model which could read a story and explain whether the main character acts morally could be useful for settling many civil and legal disagreements.

The results from this sort of text generation project could be very useful. There has already been research into and applications of AI in the legal system (Dymitruk, 2019) and the massive labeled data set on this subreddit could help train models to make legal judgements, or at the very least help settle simple arguments.

Although there are many posts on the subreddit, it appears that only some 63,000 posts are easily usable for NLP, as detailed in this article. There is some research using AITA; (Botzer et al., 2021) analyzes AITA comments to determine whether they judge "the user who made the original post to have positive or negative moral valence". This Stanford project builds on works by (Tigunova et al., 2019) to create a neural classifier for AITA posts. However, their dataset is relatively very small compared to the one in the aforementioned article. The small size of their dataset caused high model variance. With that larger dataset, this should not be an issue for training a better classifier. Reddit is a common source of training data for NLP research, so learning the API may prove useful.