Computational Empathy

Computational Empathy

A thesis submitted
by
London Lowmanstone

in partial fulfillment of the requirements for the degree of Bachelor of Arts in Computer Science and Philosophy

Harvard College
Cambridge, Massachusetts March 26, 2021

Abstract

In our current environment, computers behave in ways that are unexpectedly different from humans. However, the term “empathy” is defined in relation to human behavior, not computer behavior. Thus, until the general public becomes used to computer behavior, we should be wary of using the term “empathy” to describe computer systems, which may act in unexpected ways for something that is considered to have empathy. This point is illustrated through the creation of machine learning models which have a particular type of empathy that is only exhibited for inputs inside of their training set.

Acknowledgements

I would like to give a huge thank you to Cheryl Chen and Kai Wang for their consistent support and help throughout the thesis process. I would also like to thank Joshua D. Greene and Milind Tambe for being willing to be readers for this thesis and for helping me by providing feedback, resources, and structure. In addition, thank you to my roommates, Montague Mawere and Dimitar Karev for their advice and emotional support especially during the more stressful times. Finally, I'd like to thank my friends and family for helping me get to this point; I could not have done it without you. Thank you!

Table of Contents

Abstract **1**

Acknowledgements **2**

Table of Contents **3**

Contributions **5**

Motivation **5**

Approach **6**

The Models **8**

Motivation 8

Related Work 9

Dataset 11

The Raw Dataset 11

The Final Dataset 13

The Base Model 14

Results **15**

Evaluation 16

Train-test Split 16

Custom Accuracy Function 16

Base Model Performance 17

Layer Size 18

Decreased Layer Size 18

Increased Layer Size 19

Analysis 19

Amount of Layers 20

Decreased Layers 20

Increased Layers 21

Analysis 21

Dropout 22

Small Dropout 22

Medium Dropout 23

Analysis 23

LSTM 24

Bidirectional LSTM 25

Overall Analysis 25

Philosophy Introduction **28**

Entities **28**

What is a computer? **29**

Counterfactuals **33**

Computers - Revised **34**

Empathy **35**
- Overview of Cognitive and Emotional Empathy 36
- The Three Frameworks 36
  - The Results Framework 38
  - The Process Framework 38
  - The Properties Framework 39
  - Summary 40
- Cognitive Empathy 43
- Emotional Empathy 45
- Summary 47

Model Analysis **48**
- Metadativity 49
- Analysis (cont.) 51
- Accordativity 51
- Analysis (cont.) 52

When should we say computers have empathy? **53**
- Results-Based Definitions for Particular Scenarios 53
- Property-Based Definitions for Counterfactual Robustness 54

Conclusion **56**

Bibliography **58**

Related Works **61**

Contributions

This thesis is a joint thesis between computer science and philosophy, and thus has contributions to both fields. For computer science, this thesis introduces the first known system that attempts to identify topics that a person is emotionally sensitive to, given text that is written by that person. Note that this is different from sentiment analysis, which attempts to determine what emotions the author is feeling. Our model, instead, determines topics such that changes in that topic would cause significant change to the author's emotions. The base artificial neural network model for this task does not generalize, but we have included a quantitative analysis of modifications that may improve generalization, as well as an analysis of why neural networks in general may be difficult to train for this particular task. This thesis also provides a dataset with over one hundred paragraphs from memoirs labelled with topics that the authors appear to be emotionally sensitive to. For philosophy, this paper provides guidelines for usage of the term "empathy" when applied to computers, and situates these guidelines in a more abstract framework which can be applied beyond empathy and computers. This paper also defines a few terms that may be useful for future technologists when describing the kind of empathy that their computer systems have and provides an analysis of the machine learning models using these terms.

Motivation

Imagine you are watching a movie on your computer and the internet slows down, causing the show to freeze. This is upsetting, as the movie has just gotten to a good part, and now the magic has been broken while the movie loads. Instead of coldly displaying a loading symbol, the computer system itself becomes upset, displays a message that says "yeah, this is really annoying," and begins trying many different methods in rapid succession to try to raise the

internet speed as fast as possible. It not only understands how you feel, it feels the same way you do.

Computers today aren't like this. They just display a loading symbol, or, in special cases, generate a game for you to play while you wait. They have no idea if you are upset about the situation, and they don't really care whether or not you play the game while you wait. They are not empathetic.

We also live in a time where computers are said to possess particular traits that can mislead people into believing that the computers will behave differently than they actually do. Self-driving cars are hailed as "safe drivers" and then crash into overturned trucks on the highway (Stumpf). Siri is labelled as an intelligent system, but, as of writing, if you ask Siri "How many letters are there in the word dog?" Siri replies with "25 characters." [1] How can we go about using the term "empathy" for computers, such that it does not lead people to believe that the computer will behave in certain ways, only to be surprised (and potentially harmed) when it does not? Given that computer systems are getting better at understanding how people feel, and may someday have what we may consider to be emotions, how should we go about describing the sort of "empathy" that these intermediate computer systems have? These are the questions that this thesis sets out to answer.

Approach

It may be expected that this thesis gives a particular answer about how we should define empathy for computers. However, since the terms "computer" and "empathy" are defined extremely broadly, and we live in a society with constantly changing definitions, I have decided

[1] It appears that Siri mistakenly interprets "How many letters are in the word dog" as asking "How many characters are in the phrase 'are there in the word dog'?" That longer phrase does indeed have 25 characters (including spaces), but it is obvious to most humans that that is not what is being asked.

to instead discuss the advantages and disadvantages of several different approaches to defining empathy for computers (which I call "computational empathy"), rather than arguing specifically for one approach in all cases. As it turns out, these approaches are used for other definitions beyond just computational empathy, and so the advantages and disadvantages are quite general. Thus, we will both discuss the generalized results as well as how those ideas apply specifically to computational empathy.

Since we are discussing multiple different approaches, this thesis has multiple conclusions. The main terms will be defined in more detail later, but I think it is worth mentioning the conclusions here so that readers are aware of what we are building towards. The first conclusion is that results-based definitions are useful primarily in particular scenarios rather than as generalized traits. When applied to computerized empathy, this means that when we say "the computer is being empathetic," our definition of "empathetic" should be based on the results of the computer's behavior. The second conclusion is that process-based definitions are useful primarily for describing traits, rather than particular scenarios. When applied to computerized empathy, this means that when we say "the computer has empathy," our definition of empathy should be based on how the computer behaves in general. The final conclusion is that property-based definitions are useful for describing terms where the original objects that the term applied to have substantially different counterfactual behavior than the new objects that the term is being applied to. When applied to computerized empathy, this means that since humans behave quite differently than computers, we probably shouldn't say that a computer has empathy until people are used to how computers behave. I believe that this final conclusion is the most novel and interesting conclusion in this thesis, and therefore it is highlighted as the primary conclusion.

To build up to these conclusions, this thesis is written in the following structure. First I will introduce my machine learning model. I will analyze the success of the model and propose potential ways that the model might be improved based on experimental data. Then, I will define key terms such as “computer” and “empathy,” building the up terminology to do so along the way. We will then analyze whether the sort of empathy the machine learning model has. Finally, we will describe recommendations as to when we should say computers have empathy.

The Models

Motivation

I do not know what laws are being worked on right now. I have some idea of a key few, such as COVID relief bills, but what those bills actually contain and what is up for debate, I rarely know. From talking with politicians, friends, and family members, this appears to be a common trend. People may know key laws *after* they are passed, but very few know the bills that are currently being worked on, much less how they can become involved.

Previously, I had heard about Resistbot, an automated system that allows one to text messages that will then be sent as letters to representatives (Resistbot). However, Resistbot only enables communication from constituents to legislators; it doesn’t increase knowledge about the laws themselves.

Many states have a website that shows current bills that are in the process of becoming laws. However, most of these bills are written in legalese that’s difficult to understand for most people, and even if they were easy to understand, there’s so much contained inside of them that it is unlikely people would want to read them. People rarely read Terms and Agreements that they agree to on a daily basis, much less laws that are unlikely to impact them in a meaningful way.

Thus, my goal was to develop a system that would detect bills with provisions that would affect individuals' lives. Essentially, a person would write stories or talk to the program about their daily life, and it would detect particular topics that would have a large emotional impact on the person if they changed. The full system consisted of three separate programs. The first would gather stories from users about their life and run an encrypted or local detection of topics that would likely greatly impact the user's emotions if they were to change. Then, another program would scan through existing bills and generate scenarios of how those bills could be implemented if they became law. Finally, a third program would detect whether or not a particular implementation of a bill could cause a substantial change to a sensitive topic computed by the first program. If so, then the program would send a text to the user, notifying them about what bill could potentially impact their life, and use technology such as Resistbot to enable them to contact the correct representatives in order to share their opinion (whether positive or negative).

Obviously, this system is much too large and complex to be feasible within the given time period for this project, so instead, I implemented only the first program for detection of emotionally sensitive topics. The goal is to detect particular topics that, if those topics were modified through the implementation of a law, the author would likely have a large emotional shift.

Related Work

The field of "affective computing" was formalized in a 1995 paper by Rosalind Wright Picard, who defines it as "computing that relates to, arises from, or deliberately influences emotions" (Picard). Since empathy is concerned with the topic of emotions, analyses of computational empathy fall under this broad umbrella of affective computing.

Much progress has been made since Picard's original paper, especially in the area of textual analysis for emotional content. Such analysis, known as affective sentiment analysis, generally falls into two categories: emotion recognition and polarity detection. Emotion recognition programs output a given emotion for a particular piece of text. Polarity detection usually takes in a piece of text and outputs a value on a single axis. For example, "thumbs up" vs "thumbs down" or "like" versus "dislike" (Cambria). While these are similar to what my models are attempting to accomplish, the outputs of these models are either fine (as with emotion recognition) or coarse (as with polarity detection) *emotions*, whereas my model outputs *topics*. Thus, while sentiment analysis programs might eventually be able to aid in the task that I am setting out to solve, of detecting topics that people are emotionally sensitive to, sentiment analysis programs themselves will not immediately solve this problem.

Another avenue of similar work is topic detection and modelling. Topic detection and modelling algorithms take in text and output the main topics of that text, either as singular words or as groups of words that form a topic. Common algorithms for this task include Term Frequency-Inverse Document Frequency (TF-IDF), Latent Dirichlet Allocation (LDA), and Latent Semantic Analysis (LSA) (Blei; Foltz; "What Is TF-IDF?"). While each of these algorithms is a method of determining topics found in text, the goal is to find the most *important* topics in the text, rather than topics that people are *emotionally sensitive* to. Thus, if the important topics have little to do with the author, or are not things that the author is emotionally sensitive to, topic detection and modelling will still choose them as the most important terms.

For example, consider the following paragraph from *The Price of Black Ambition* by Roxane Gay:

> Du Bois was a vocal proponent of the "Talented Tenth," this idea that out of every ten black men, one was destined for

`greatness, destined to become the powerful leader black people needed to rise up and overcome and advance. This 10 percent of men were to be educated and mentored so they might become leaders, the front line for much-needed sociopolitical change.` Here, Gay is merely stating facts and describing another person's beliefs. This paragraph alone does not suggest that Gay is emotionally sensitive to any of these particular topics. While the TF-IDF algorithm (which is primarily based on word frequency) correctly identifies the main terms as "black," "men," and "destined," these terms are not topics that the author should be considered emotionally sensitive to, given the content of the paragraph.

Thus, while affective sentiment analysis, topic detection, and topic modelling are helpful tools, none of them are suitable for the task of detecting topics that people are emotionally sensitive to. Sentiment analysis determines emotions, rather than topics, and topic detection and topic modelling disregard perspective and emotions. So, a new model is needed.

Dataset

The Raw Dataset

The raw dataset consists of four memoirs located via Early Bird Books, each in a text file. Each paragraph within the memoir is considered individually and without context and is annotated for topics that I considered the authors to be emotionally sensitive to, given that single paragraph. Altogether, the raw dataset is four files that follow the format of one paragraph of text from the memoir, followed by an empty line, followed by topics with one topic per line, followed by one or more empty lines, followed by the next paragraph from the memoir.

My rule for determining what topics the authors were emotionally sensitive to given a particular paragraph was the following: "if something made a substantial change to X (or how X

behaves), it is likely that the author's emotions would substantially change," where X is a given topic. For example, here is a paragraph from a memoir titled "Age Appropriate" by Jen Doll:

> After we loaded the rental car and I dutifully fastened my seatbelt in the backseat, assuming the position of so many family road trips past, I realized I hadn't mailed my maintenance for my Brooklyn apartment. "Hang on — I'll be right back!" I yelled, grabbing the envelope with the check in it and dashing across the street toward a mailbox. My dad waited at the side of the road, but then came a surge of traffic, and then a cop, and he had to drive on. "Noooooooo!" I yelled, chasing after the rental car (what kind was it anyway? I had no idea!) in the heat, knowing even as I did my perfunctory sad jog that there was no way I'd catch up.

From this paragraph of memoir text, I determined that the topics the author was emotionally sensitive to were as follows. "Maintenance," since if she didn't need to pay the maintenance, she wouldn't have been so stressed; "Brooklyn," since places where people live tend to have a large impact on their emotions; "traffic," since if there was less traffic, she would not have lost the car; "cops," because if the cop had not been there (or been more understanding), her dad might have been able to stay for longer, and "speed," which is a one-word version of "traffic speed," which was chosen since, if the traffic speed had been slower on that road, she might have been able to catch her father's car. I assumed that if she did not end up losing the car, her emotions would have changed significantly to not be so distressed and then sad.

There are a few quirks in the raw dataset. Since dialogue tends to be one paragraph per line of dialogue, lines of dialogue next to each other are conglomerated into a single paragraph in

the dataset. In addition, the raw dataset has topics which may be difficult for the computer to detect. I annotated these topics to begin with an asterisk ("*"), and the program that converts the raw dataset to the final dataset that is used for training will ignore topics that begin with an asterisk. I also used the asterisk to annotate paragraphs that had no topics. These paragraphs are annotated with a single ignored topic, "*(None)", which, as described previously, will be ignored in the final dataset because it begins with an asterisk. Finally, all smart quotes in the original memoirs were replaced with straight quotes in the raw dataset. The raw dataset is located in the appendix.

The Final Dataset

The final dataset is created via a python program which reads through the raw dataset and generates a database for training. The final dataset is different from the raw dataset in a few key ways: 1. The final dataset does not contain any topics that begin with an asterisk 2. Any multiple-word topics (such as "traffic speed") are ignored 3. All lines beginning with a hashtag in the raw dataset are ignored 4. All paragraphs have punctuation removed them 5. All words in the paragraph are put into lowercase 6. All paragraphs that do not have any valid topics are ignored. In addition to these differences, it is common for all of the inputs to the neural network to be of the same size. Thus, the final dataset pads the inputs, adding on the word "xxx" to the end of smaller paragraphs until the word length of each smaller paragraph reaches that of the largest paragraph (which contains 241 words).[2] The resulting final dataset, with 115 annotated paragraphs, can be represented as a spreadsheet, with one column containing the words in the paragraph, and another column containing the words describing the topics associated with that paragraph. The final dataset is located in the appendix.

[2] Technically, the final dataset uses the Python keyword `None` to pad the inputs, but these `None` values are converted to "xxx" before being embedded by the FastText model.

The model assigns a unique representation to each word in the final dataset. (This representation takes the form of a one-hot vector.) So, capitalized topics would be considered different topics than non-capitalized topics (even if they were the same word), and topics with multiple words would be considered as unique words as well. Thus, the final dataset is limited to singular lower-case words.

The Base Model

For the purposes of this thesis, I have created multiple models which attempt to solve this task of detecting topics that people are emotionally sensitive to from text that they have written. However, there is one base model which all of the other models are compared to. The Tensorflow summary of this model is provided in the following image:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 241)]             0
_________________________________________________________________
embedding (Embedding)        (None, 241, 300)          748200
_________________________________________________________________
reshape (Reshape)            (None, 72300)             0
_________________________________________________________________
dense (Dense)                (None, 3000)              216903000
_________________________________________________________________
dense_1 (Dense)              (None, 3000)              9003000
_________________________________________________________________
dense_2 (Dense)              (None, 2494)              7484494
=================================================================
Total params: 234,138,694
Trainable params: 233,390,494
Non-trainable params: 748,200
```

Overall, the base model has five layers (not including the reshape layer). The first layer takes in a paragraph. This paragraph is represented by 241 indexes. Each index refers to a particular word. Then, the indexes are used to find the word, and the word is passed to the

second layer for embedding.[3] The second layer uses a model called FastText that was trained on a language modelling task on Wikipedia data; it embeds words into a vector with 300 dimensions (Bojanowski).[4] This is followed by two feedforward layers (also known as "dense" layers) with 3,000 units each. The final layer is another feedforward layer with one unit for each word in the dataset, resulting in the final layer having 2,494 units. A softmax activation function is applied to the final layer, so each of the 2,494 output values will be between 0 and 1 inclusive. The targets are represented as multi-hot vectors, so a value of 1 at an index in the final layer indicates that the word corresponding to that index is a topic that the person is emotionally sensitive to.

FastText is ideal for embedding text from memoirs as it embeds words using subword information. Unlike GloVe and word2vec embeddings, FastText allows us to embed words that were not seen in the original Wikipedia dataset (Mikolov; Pennington). This is important if we want to be able to have our model deal with new words in new environments that were not encountered in Wikipedia. In other words, we would like our model to work for people who don't always use language that is in Wikipedia. Thus, FastText provides an advantage over other common word embeddings such as GloVe and word2vec.

Results

We will now evaluate the impact of four different architecture changes to our base model. 1. The size of the feedforward layers. 2. The number of feedforward layers 3. Adding dropout layers between the feedforward layers 4. Replacing the first feedforward layer with a long short-term memory (LSTM) layer and 5. Replacing the first feedforward layer with a bidirectional LSTM layer.

[3] Technically, the embeddings are precomputed and the index is used to look up the embedding, but the process is essentially the same

[4] You can think of this as representing a word as a list of 300 numbers

The decision to test adding dropout, using an LSTM layer, and specifically a bidirectional LSTM layer, was partially motivated by Kratzwald et. al., who created a neural network for emotion recognition (Kratzwald).

Evaluation

Each model is trained for 100 epochs on the entire training set and then is evaluated on the test set. Throughout training, we monitor both the loss and accuracy metrics during each epoch. Details and explicit values during training can be found in the iPython Notebooks for each of the training sessions. For most of the models where it seemed possible that interesting behavior might occur after 100 epochs, I continued to train the model in order to verify that no interesting behavior occurred. This was indeed the case for each model I tested, so I will only be presenting the results for each model from the first 100 epochs of training, as the patterns that occur within that time period are likely to continue to hold in future epochs as well.

Train-test Split

The dataset of 115 examples is split into 92 examples (80%) that are used for training and 23 examples (20%) that are used for testing. The split was performed randomly by a function from the Scikit-learn package, using a random seed from within the range [0, 100] (Pedregosa). The same dataset and split was used to evaluate all of the models

Custom Accuracy Function

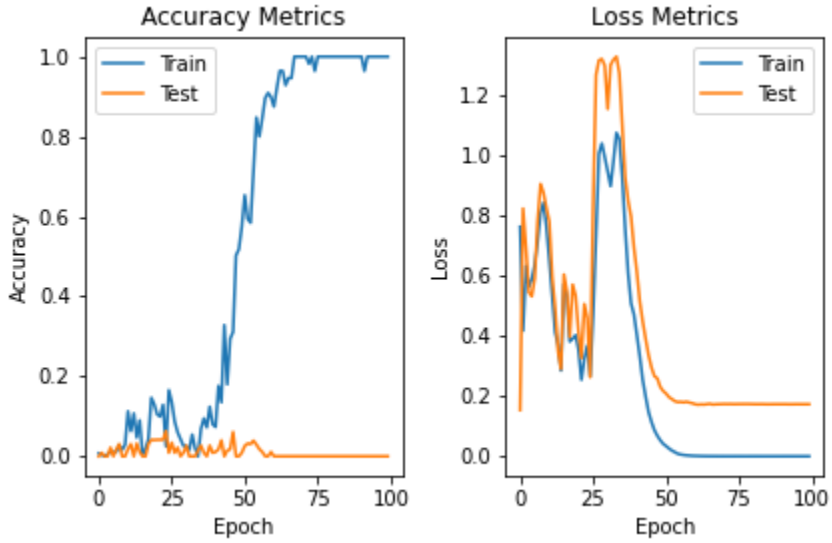
While the network trains by using a binary cross-entropy loss function, we have also defined a custom accuracy function to more accurately determine the success of the network.

This custom accuracy function is defined as follows. For a given example consisting of a paragraph and n topics, we look at the top n words that the model predicted. For each of the n

topics that are in the top *n* words that the model predicted, the model receives 1 point. Then, we give a total score for the example by dividing the points earned by the total possible number of points, *n*. Thus, a perfect score is 1, where the network correctly predicts all of the topics within its top *n* predictions, and the worst possible score is 0, where the network fails to have any of the topics within its top *n* predictions. Overall, the accuracy function for each example is merely the percentage of how many topics the model "got right." Then, the final score for the model across all examples is merely an average of the scores for each example. Thus, the final score also ranges from 0, where the network failed to predict any topic in any example, up to 1, where the model correctly predicted every topic in every example.

Note that the definition of this accuracy function means that for examples which only have a single topic, the model can only score a 0 or a 1; it cannot receive a score of 0.5 if there is only a single topic for the example. Thus, for examples with small amounts of topics, there will be sudden large changes in the accuracy as the model improves, rather than the more gradual improvement as one might expect from a loss function.

Base Model Performance

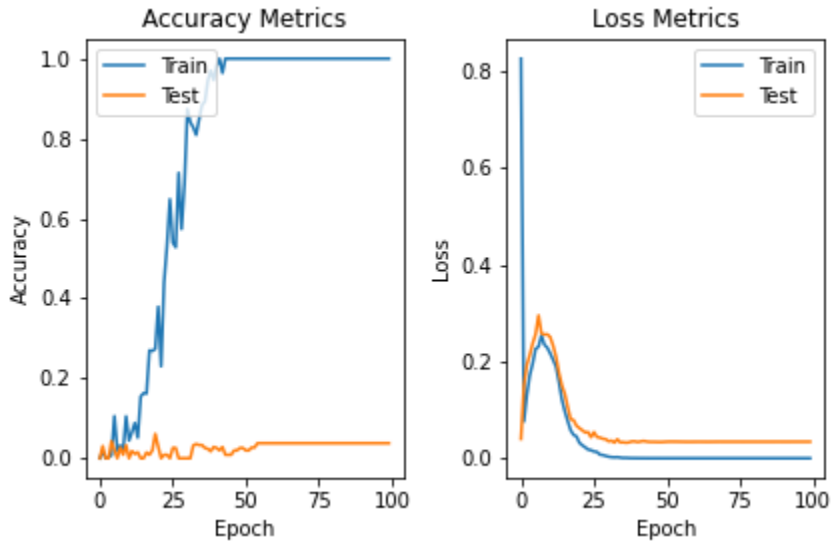


As we can see from the diagram, our base model suffered from extreme overfitting. It was able to accurately predict each of the topics from each paragraph in the training set, but was unable to determine any of the topics correctly in the test set by the end of its training. The model generally stabilized in terms of loss and accuracy at around epoch 69, which is when it achieved perfect accuracy on the training set for the first time.

Layer Size

Our base model has two hidden feedforward layers of 3,000 units each. We experimented with changing the number of units per feedforward layer by lowering the number of units to 1,500 each and then raising the number of units to 4,500 units per feedforward layer.

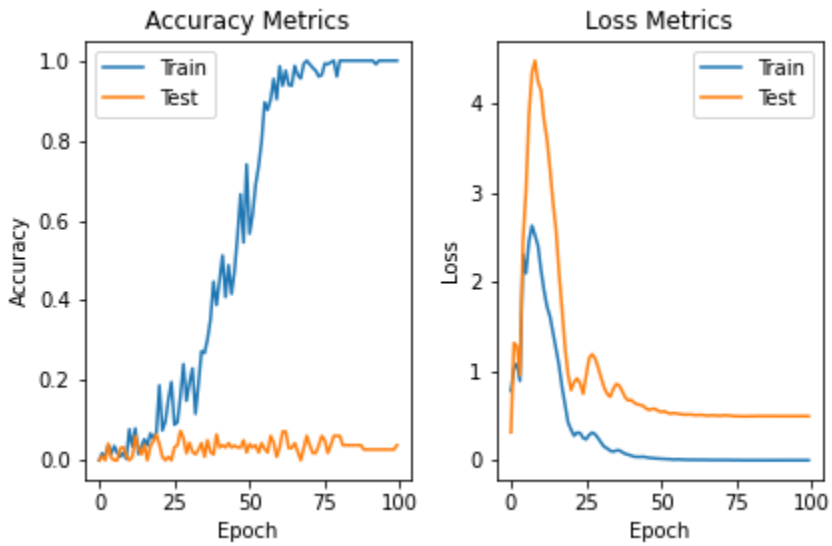
Decreased Layer Size



Decreasing the layer size to 1,500 units for each of the two hidden layers appears to have drastically improved the performance of the network. Not only does it cause the network to obtain 100% accuracy on the training set within a mere 45 epochs, our final model generalizes

slightly to the test set and stabilizes at a final test set accuracy score of 0.038. In other words, its average score for examples in the test set was 0.038.

Increased Layer Size



Increasing the layer size to 4,500 units seemed to increase the generalizability of the network up to 0.038 accuracy on the test set, but, in turn, also caused the model to only reach 100% on the training set by epoch 79, a full 10 epochs later

Analysis

The main factor that changing the layer size appeared to impact was the training convergence time. With a small number of inner units, the model converged within 45 epochs, in comparison to the larger network which only converged by epoch 79, with no noticeable loss in accuracy between the two models.

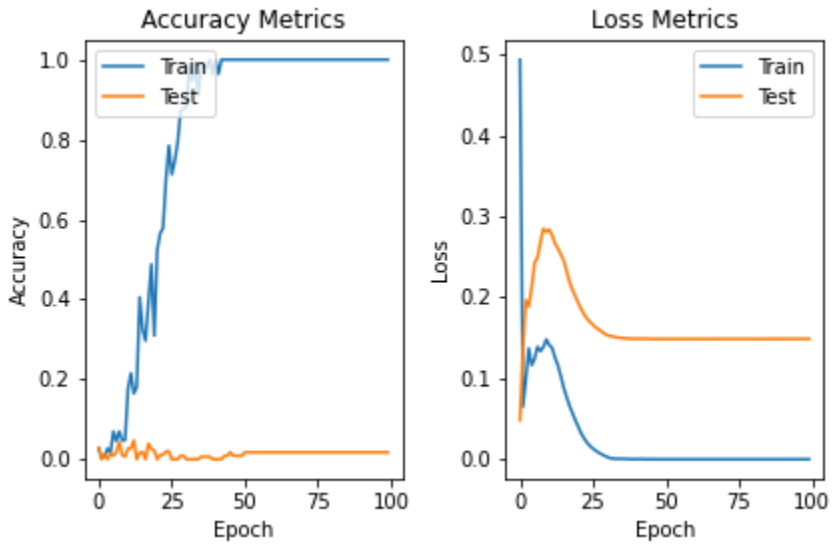
Since both increasing and decreasing the number of units in the hidden layers both led to increases in generalizability, it may be that either 3,000 units was a particularly poor choice of

units for the base network, or that the upgrades in generalizability are so small that they are merely due to chance.

Amount of Layers

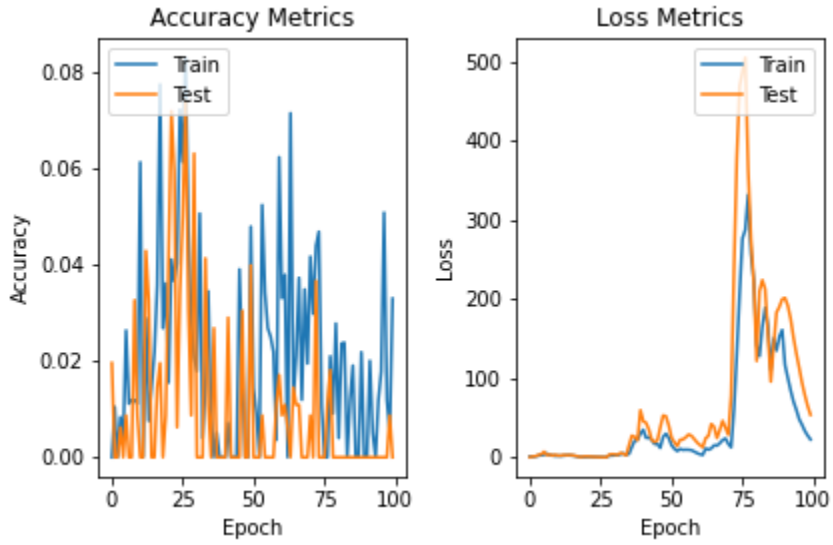
Our base model has two hidden layers, each with 3,000 units each. We experimented with removing one of the layers, as well as adding another layer with 3,000 units.

Decreased Layers



Decreasing the layers caused a decrease in the time to converge, with the test accuracy settling at epoch 50 and the training accuracy settling at epoch 47, but only a very slight increase in generalization, up to 0.017 accuracy on the test set.

Increased Layers



Increasing the number of layers to three hidden layers caused the network to fail to converge within a reasonable amount of time and score extremely poorly in terms of accuracy and loss by the end of its training. However, this was the only network to output words during the testing period that were not found in its training set. Unfortunately, none of these words were correct, as its test accuracy was 0, matching that of the base model.

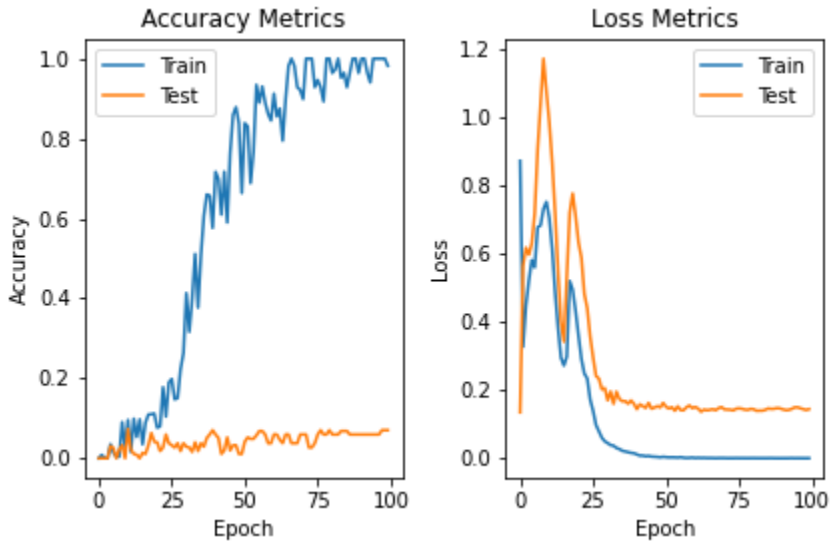
Analysis

Decreasing the number of layers seems to have improved both convergence and generalizability. It may be that the larger network did not have enough time to train, but it also did not look likely that even with much time to train that the network would greatly increase its accuracy beyond that of other models. Overall, it looks like the base model could have been improved by removing a layer.

Dropout

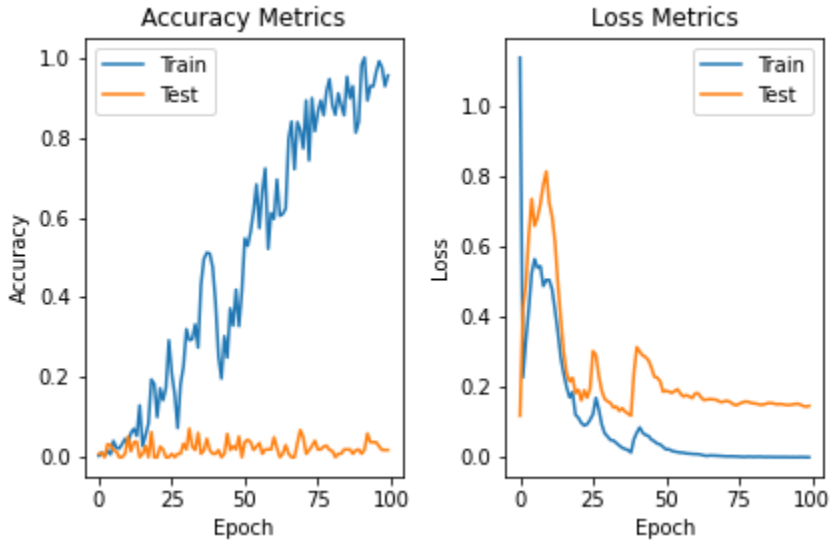
The base network did not contain any dropout layers. We experimented with adding two dropout layers, one before each hidden layer. We tested setting the dropout rate to 0.25 and 0.5 respectively.

Small Dropout



Adding in two dropout layers with rates of 0.25 before each of the hidden layers appears to have greatly raised generalizability up to an accuracy of 0.07, the highest accuracy we have seen thus far. However, it appears to have done this at the cost of a longer training time, with the model failing to cleanly converge after 100 epochs.

Medium Dropout

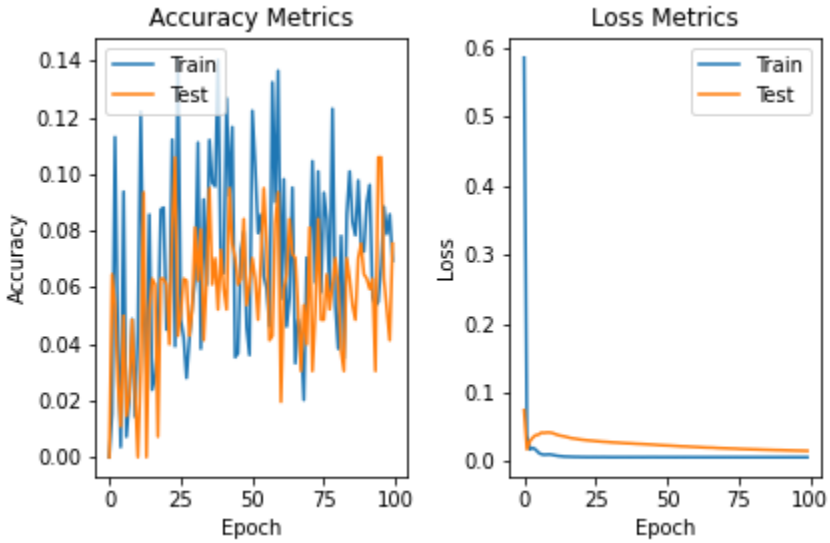


Adding in two dropout layers with rates of 0.5 before each of the hidden layers appears to have merely raised the convergence time without much gain in generalizability. The maximum accuracy achieved during all of training was 0.068 at epoch 70, which dropped to 0.018 by the end of training.

Analysis

Overall, adding small amounts of dropout may greatly increase the generalizability of the model at the expense of taking longer to train. However, increasing the dropout size too far may cause the model to begin decreasing in generalizability.

LSTM

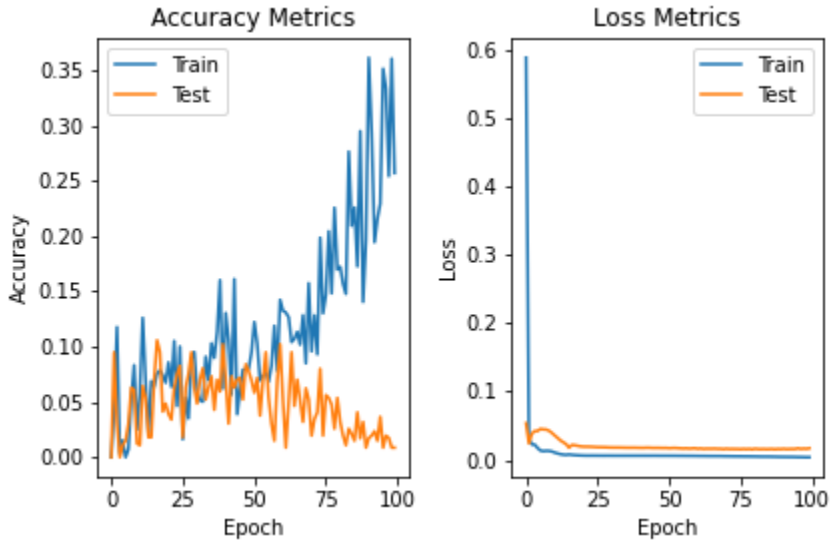


Replacing the first layer of the base model with an LSTM layer of 500 units led to the model to not converge with respect to accuracy. However, the accuracy remained relatively high on the test set in comparison to other models, ending at a value of 0.0755 accuracy on the test set, which was up to 0.106 as recently as epoch 96.

Despite the high accuracy on the test set, the extremely low accuracy with respect to the training set suggests that this model has not truly found consistent patterns in the data.

In addition, training the LSTM took nearly twice as long as other models. Each epoch for non-LSTM models generally took 2-3 seconds, whereas the LSTM model took 3-5 seconds to train for each epoch.

Bidirectional LSTM



Replacing the first layer of the base model with a bidirectional LSTM layer of 500 units caused the model to begin converging towards a fairly low test set accuracy. However, unlike the unidirectional LSTM, this model's accuracy on the test set appeared to be increasing, rather than failing to converge. The test set accuracy at the end of training had fallen to 0.009 and was not on an increasing trend.

Overall Analysis

There was much diversity in behavior in the different models, from failing to converge while having relatively high test accuracy to converging extremely quickly with low test accuracy. The only modification which seemed to offer clear benefits was adding a small amount of dropout before each of the hidden layers.

However, there is one core similarity between all of these different variations on the original base model: all of the models overfit and did not end up adequately generalizing to examples outside of their training set. In fact, only one model, the model with an added

feedforward layer, predicted any words that weren't found within the training set. However, none of its predictions on the test set were correct by the end of training.

Initially, I had hoped that the neural networks would find patterns within the words and general indicators that would allow the network to confidently predict new words as topics, due to their surrounding context. However, it seems as though within this small dataset, each of the networks essentially trained to memorize the answers for the examples in the training set, and then merely predicted those exact same topics for examples in the test set. If we removed all of the topics from the test set that were not found in the training set, and a model was randomly guessing topics from the test set, the random model would have only a ~32% chance of getting a non-zero score. This suggests that our models that do generalize, however slightly, are performing better than random. However, since our accuracy function looks at the top *n* predictions for each example, and removing non-training topics from the test set reduces that *n*, the chance of getting a non-zero score may be much higher, since our models are essentially allowed an "extra chance" to predict a training topic for each of the non-training topics that are in the test set. Thus, it may be that our models are only generalizing as well as a model that randomly guessed topics found in the training set. A model that only predicted words from the training set topics could achieve an accuracy of up to 41%, whereas the highest accuracy among the models built was merely 7%.

It may be that due to the backpropagation algorithm, neural networks are more likely to converse on weights that cause them to only predict outputs in the training set, as doing so never penalizes them. However, in order to complete this task effectively, a model will need to be able to train to recognize patterns in the dataset confidently enough to predict words that never appear in the training set in order to score well on the test set.

Altogether, most of these models were able to achieve 100% accuracy on their training set, but did not generalize much beyond that, with the most successful stable network only achieving an accuracy of 7% on the test set.

Philosophy Introduction

We have now analyzed the success of the machine learning model from a quantitative standpoint. But what about from a philosophical standpoint? Does the base model or any of the other models we trained exhibit empathy? If so, what sort of empathy?

To do this, we will define a few key terms that are needed to understand the answer to this question such as “computer,” and “empathy,” and we will also define useful terms such as “entity,” and “counterfactual” along the way. In addition, we will establish three frameworks for categorizing different definitions of empathy. These definitions will then allow us to evaluate different responses to the general question of when computers have empathy. After having evaluated those responses, we then turn to the question of whether our particular program has empathy. As we find out, none of our models have any usual form of empathy, and we end up defining the terms “metadative” and “accordative” in order to describe the form of empathy that our (better) models exhibit. Finally, we conclude with an analysis of when particular definitions for empathy may be useful or misleading.

Entities

Before we dive into the definition of a computer, a note of my usage of the word “entity” is needed. Throughout this thesis, I use the term “entity” to refer to general things that portions of this thesis may apply to, even if they don’t neatly fall into the boxes of “computer” or “person.” Since the goal of this thesis is to discuss the empathetic capabilities of computers, many of the terms used will apply to both humans and computers and may also be applied to animals or beings that consist partially of computer components and biological components (often called “cyborgs" when referring to beings with human biological components). While some definitions of empathy have been applied to animals, others argue that particular aspects of

empathy may only take place in humans (Decety). One might consider a spectrum of consciousness where humans, who are almost always considered to be conscious, are located on one end, animals and cyborgs, of which some are thought to be conscious, are somewhere in the middle, and computers, which are often considered to not be conscious, on the other end. The term "entity" covers this entire spectrum. However, since the future of research is unknown, it is possible that many of the aspects of human empathy could eventually be discovered in animals or arise in computing or cyborgs or groups thereof. Due to the uncertain nature of the future of computing, any attempts to limit the definition of empathy may limit the applicability of this work to future environments in which the consensus on the capabilities of computers has shifted. Thus, by using the general term "entity," this thesis attempts to establish a broad guideline for when any entity should be considered to have empathy.

What is a computer?

In many cases, computers are defined in terms of Turing Machines. *Turing Machines* are defined by the following components:

1. An infinite number of slots called a "tape," each of which contains a single symbol.
2. A "head" which can read and write to the tape
3. A set of states that the Turing Machine can be in.
4. A set of rules (called "transitions") that determine what symbol the Turing Machine writes to the tape and which direction it moves along the tape to read the next symbol, given its current state and the symbol it is currently reading.

To use a Turing Machine, one writes symbols on the tape to describe the problem that one wants to solve, and then sets the Turing Machine to the start state and has the "head" begin reading the

first symbol on the tape and follow its given rule set from that point onwards. When the Turing Machine is done computing the answer, it will have written the answer in a predetermined place on that tape.

In general, the class of Turing Machines identifies a category that consists of abstract entities. Each individual Turing Machine has particular states and transitions that are defined for that Turing Machine. Two Turing Machines are trivially identical if they have the same states and transitions. That is to say, individual Turing Machines are not defined by the contents of their tape. To *program* a Turing Machine is merely to define a particular Turing Machine by specifying the states and transitions of that Turing Machine.

All Turing Machines are completely deterministic, and are only able to compute functions. That is, a particular Turing Machine will output the exact same output every time it is run on a particular input, and do so in the exact same manner every time. In addition, if a particular input could have more than one possible correct answer, a Turing Machine will only be able to generate one answer. (Though this answer may list all the correct answers, assuming there are a finite number of them.)

Turing machines are generally used as a model for *computability*. That is, Turing Machines define the limits of what questions all computers (including quantum computers) could eventually determine the answer to. When using this definition, it is important to note that a Turing Machine has infinite and perfect memory; a Turing Machine's tape is consistent and infinitely long, and Turing Machines can take as long as needed to compute a result. (This can be thought of as Turing Machines being able to have infinite time to compute a result, though this conception is not quite correct, since if a Turing Machine actually *used* an infinite amount of time to compute a result, it would never complete its computation. Thus, the previous

description, which does not rely on the term “infinite time,” is more precise.) This means that many computations that seem extremely complex or difficult will still fall under the capability of a Turing Machine. For example, a Turing Machine could compute a perfect physical (Newtonian) model of the universe and output exactly what would occur during the next millenia, if given the input of the current state of the universe.

Since Turing Machines are so powerful, if our definition of “computer” was limited merely to Turing Machines, there would be nothing in real life that we would consider to be a computer. This is obviously not the case. Thus, we will define computers as Turing Machines with reduced capabilities. Our initial definition is as follows: an entity is a *computer* if it computes some subset of the things a Turing Machine can and a Turing Machine can compute all of the things that the entity computes. This means that the most powerful computers are Turing Machines, and what we consider to be computers are merely computationally weaker versions of Turing Machines.

Given this odd definition in terms of Turing Machines that have infinite and perfect memory and as much time as needed to run, one may think that computers should be defined in much more simpler, relatable terms. For example, one could define computers as entities which carry out sequences of logical operations in order to transform inputs into outputs. This would indeed be a useful definition of computers, but then would require a further explanation of what one considers to be “logical operations.” By defining computers in terms of Turing Machines, we establish more precisely what computation is. Under our current definition, a “logical operation” would in fact be the operations that a Turing Machine is able to carry out.

Some of the computations that Turing Machines are capable of are extremely simple to carry out. For example, consider a system consisting of marbles and a tube. One can consider the

marbles that one puts into the tube to be the input, and the marbles that one gets out of the tube to be the output. Then, this marble and tube system can compute the identity function; whatever input is put into the system, the system returns back out. If one puts in a blue marble, one gets a blue marble back out. Turing Machines, trivially, can perform the identity function of copying the input section of the tape into the output section of the tape. Thus, the marble and tube system has some computational power and satisfies the first part of the definition of a computer. In addition, if one believes that this identity function is the only computation the marble and tube system can perform, then, since this function can also be computed by a Turing Machine, the marble and tube system is in fact a computer.

Given that the first condition of the definition of a computer is so trivial to satisfy, it may be worrying that the second condition seems to merely amount to simulation. That is, if a Turing Machine can simulate everything about an entity, then, by definition, the Turing Machine can perform every computation that entity can perform. So, if we assume that the entity can do anything that would be considered to be computation (which is a rather simple assumption, as described in the marble and tube example), then that entity would be a computer, merely because it is simulatable. It seems natural to think that, surely, just because something is predictable / simulatable / computable does not mean that the thing itself is a computer. However, for our *computational* definition of computers, the thing itself is indeed considered to be a computer. The entity performs the exact same functions, or one could say, carries out the exact same computations, as a computer, and, therefore, under our definition, it *is* a computer.

Thus, according to our definition of computers, if one believes that humans are defined by their actions, mental processes, and experience, and it turns out that human actions, mental processes, and experience are simulatable by a Turing Machine, then, by the Turing Machine

definition of a computer, a human would be a computer. This is because humans easily satisfy the first condition of the definition; we can definitely do some of the same operations a Turing Machine can do (such as addition). So, then, if humans are defined by aspects of ourselves that are computable, which by definition, means a Turing Machine can eventually determine them, then humans themselves would be computable, and therefore we would be computers. For example, if one believes that all human actions are predetermined, that given an exact model of one's current situation, one can predict exactly how a human will act, then one might believe that human actions are computable. If, furthermore, one believed that humans were defined solely by their actions (with no regard to mental processes or experiences), then one would believe that humans were computers under our definition of computers. We would merely be computers of our own behavior; our behavior would be the output of that computation.

Thus, we have defined computers in terms of Turing Machines. If an entity can perform any computation that a Turing Machine can, and a Turing Machine can carry out all of the possible computations that the entity can, then that entity is, under our definition, a computer.

However, one point of clarification still needs to be made. What if a Turing Machine can do all of the computations that an entity ever does, but the Turing Machine can't carry out all of the possible computations that the entity could? Is such an entity still considered to be a computer? To clarify this, it helps to introduce a new term: counterfactuals.

Counterfactuals

Within this thesis the term *counterfactual* refers to a way things could have been or a way things could be.[5] For example, if one is discussing how a student could have performed better on an exam, the counterfactual is the student performing better on the exam.

[5] This is the consequent of counterfactual conditionals

In metaphysics, counterfactuals come into play as to how we define what particular things are. The main idea is that we may define things not based on how they currently behave, but instead based on how they would behave in particular circumstances. The classic example for this is a statue made of clay. Note that the statue and the clay behave the same; it seems odd to consider the statue to be different than the clay, since the statue is made up of the clay. However, counterfactually, you can determine that they are different things via the following reasoning. If you were to smash the statue, you would destroy it. However, the clay would still exist; it would merely be squashed, not destroyed. Thus, in the counterfactual, where the statue/clay is smashed, it is clear that the two must be different things, since one would cease to exist while the other remained. Thus, things are not always defined solely by how they currently exist, but also based on how they would behave in particular situations, even if those situations never actually happen.

Computers - Revised

With the term "counterfactual" defined, we can now approach the previous question: what if a Turing Machine can do all of the computations that an entity ever does, but the Turing Machine can't carry out all of the possible computations that the entity could? This is a counterfactual concern. In this scenario, we are not focused on just what the entity is or how the entity does behave (because in that case, our previous definition determines that it is a computer), but instead on how the entity could behave.

Our definition of a computer is based on Turing Machines. So, when an entity is labelled as a "computer," one reasonably expects the entity to behave similar to Turing Machines. And, by definition, the entity does behave similar to Turing Machines. However, it also has particular capabilities (that are not put to use) that are beyond the capabilities of Turing Machines. Similar to how a Swiss Army knife is "not just a knife," such an entity is not "not just a computer." Thus,

it makes sense to say that such an entity is in fact a computer, though saying that such an entity is a computer without any other qualifications is counterfactually misleading.

So, our original definition remains correct, that if an entity computes some subset of the things a Turing Machine can and a Turing Machine can compute all of the things that the entity computes, that entity is a computer. But in situations where the entity has unused capabilities that a Turing Machine cannot simulate, we should mention that such an entity is not "just a computer."

Empathy

Now that we have defined what a computer is, we are one step closer to answering the question "when should we say that computers have empathy?" But first, we must define empathy. From the Encyclopedia of Social Psychology, there are two major types of empathy: cognitive empathy and emotional empathy (Baumeister). First, I will give a general definition of both cognitive empathy and emotional empathy. Then, I will describe three definitional frameworks that will allow us to categorize specific definitions of cognitive empathy and emotional empathy as either *results-based*, *process-based*, or *property-based*. We can then apply these frameworks to analyze different classes of definitions of cognitive empathy and emotional empathy. Overall, this section provides an answer to the question "can computers have empathy?" under various definitions of empathy. We then can evaluate which of the machine learning models may be considered to have empathy. We then will have developed the tools to analyze our machine learning models to determine under what definitions they may be considered to be empathetic.

Overview of Cognitive and Emotional Empathy

Empathy generally consists of two abilities: 1. Cognitive empathy: being able to determine how another entity feels and 2. Emotional empathy: feeling the same way that that other entity feels. If an entity does both of these things, we say that the entity has empathy. However, it's not precisely clear what it means to "do both of these things." If an entity randomly guesses how someone feels and happens to get it right, is that cognitive empathy? What if an entity is usually considered to have cognitive empathy, but now they are in a new situation and are often bad at determining how others feel — do they still have cognitive empathy? What about if an entity appears to have emotions, but it doesn't have a biological body — can such an entity have emotional empathy? Such questions have different answers based on how one defines empathy. There are three frameworks, which we will now introduce, which help to categorize differing definitions and answer such questions for broad classes of definitions.

The Three Frameworks

The Results Framework, Process Framework, and Properties Framework are three frameworks that are useful for categorizing and describing different ways of defining terms. The frameworks are best introduced through a well-known thought experiment called the Chinese Room, created by John Searle (Searle). While we are focused on defining the term "empathy" as the main subject of this thesis, the Chinese Room thought experiment is primarily focused on defining the term "understanding." However, the three frameworks work well for categorizing definitions, regardless of whether the term being defined is "understanding" or "empathy."

The usefulness of having such frameworks comes from being able to establish patterns between arguments of differing definitions. As we shall see, definitions under each framework have particular merits and demerits that make them appropriate in certain cases. Since there is no

well-defined and agreed-upon definition of empathy for all entities, having such frameworks allows us to anticipate strengths and weaknesses of future definitions. Being able to anticipate these pros and cons will, hopefully, speed up the process of reasoning and communicating about empathy in the future and allow us to find better definitions more quickly.

To introduce the three frameworks, we begin with (a slightly modified version of) the Chinese Room thought experiment, which is as follows. There is a man in a room who does not understand (in the usual sense) any dialect of Chinese whatsoever. If you were to talk to him in Chinese, he would not understand you. If you were to write to him in Chinese, he would not understand you. He doesn't know a single word of Chinese, much less anything about its syntax or characters. If you were to give him Chinese, all he would see is squiggles.

However, this man does speak and understand English, and he's given an instruction manual in English. The instruction manual enables the man to look up the "squiggles" based on their shape and tells him what to write down. He then can then give back whatever the instruction manual told him to write as a reply. Thus, one can hand the man Chinese and receive a reply. As it turns out, what the man hands back is always a reply in perfect Chinese. One can ask questions, have conversations, argue, et cetera, all by passing notes back and forth, with the man looking up the characters and following the instructions on what to write back.

John Searle then makes the following argument: "if the man in the room does not understand Chinese on the basis of implementing the [instructions] for understanding Chinese then neither does any other digital computer solely on that basis because no computer… has anything the man does not have." Thus, the core question is: what does it mean to understand Chinese? There are three main frameworks for responding to this question.

The Results Framework

The first framework, called the Results Framework, is that whether or not the man understands Chinese is based on the results that the man is able to achieve. Since the man is able to respond adequately to prompts in Chinese, under this framework, the man does indeed understand Chinese. However, if the man makes a small error when carrying out the instructions, or the instructions have a mistake in them, both cases which lead to an invalid or nonsensical response, then, some definitions under the Results Framework may claim that the man does not understand Chinese.

Definitions under the Results Framework only care about the results. They do not distinguish between different processes that achieve the same results or what things are getting those results. Any definition that falls under this framework we call a *results-based* definition.

Results can be counterfactual results in addition to real results. For example, imagine that instead of the man following instructions, the man was just guessing and drawing what to him appeared to be random scribbles, but, to the people who read his reply, are Chinese characters that form valid responses. Without counterfactuals, results-based definitions would have to consider the man to understand Chinese because he obtains the results of understanding Chinese. However, since we have defined the man to be "guessing," it is reasonable to assume that it is possible for the man to send back a scribble that is not Chinese at all. Since this is a possible result, some results-based definitions will claim that the man does not understand Chinese, based on the counterfactual results.

The Process Framework

Other definitions of "understanding" state that there's more to understanding than just the results. When using the Process Framework, whether or not the man understands Chinese is

based on the process the man follows. If the process that the man follows is a process that one would consider to constitute an understanding of Chinese, then the man understands Chinese, even if he often makes mistakes or the instructions are often wrong. It may be that results play a part in judging what processes count as "understanding"; if the man makes mistakes too often, or the instructions are wrong too often, then it may be determined that the process does not count as understanding. However, this cannot be the only consideration, or else the definition falls under the Results Framework. The Process Framework is particularly useful in situations where the man only gives correct replies, and it is not possible for him to give false replies, and yet the process he follows is not considered to be one that would enable him to understand Chinese. For example, if one believes that the man is such that he never does and never could make a mistake when reading the manual, and yet the man still does not understand Chinese, then it is likely that one's definition of "understanding" is *process-based.*

The Properties Framework

The final framework, and the one under which much debate takes place over the Chinese Room, is called the Properties Framework. Under the Properties Framework, there is something about the properties of objects in the situation that determines whether or not the man understands Chinese, and these properties are not merely about the process the man follows or the results he produces. To understand the Properties Framework, it helps to modify the thought experiment a bit. In the modified scenario, we have a woman who does understand Chinese (in the usual sense). She's a native speaker of Chinese and understands what people say and how to reply. Then, following a line of thinking popularized by Chalmers, but originally brought about by Pylyshyn, one might be able to replace a small portion of her brain with a machine that functioned in the exact same way as her brain. As Pylyshyn puts it: "If more and more of the

cells in your brain were to be replaced by integrated circuit chips, programmed in such a way as to keep the input-output function each unit identical to that of the unit being replaced, you would in all likelihood just keep right on speaking exactly as you are doing now" (Cole). The view that even with the circuit chips replacing portions of the person's brain, the person would still understand Chinese, is a view under the Process Framework (or Results Framework). Since the chips are carrying out the same process and are now a part of the person, the person now understands Chinese. However, a view under the Properties Framework might come to a different conclusion, as Pylyshn does: "You would in all likelihood just keep right on speaking exactly as you are doing now except that you would eventually stop meaning anything by it. What we outside observers might take to be words would become for you just certain noises that circuits caused you to make" (Cole). Under this view, even though the circuits are completing the same process, and the circuits are a part of the person, Pylyshn does not believe that the person retains understanding. Thus, Pylyshn's definition of "understanding Chinese" falls under the Properties Framework. Definitions under the Properties Framework argue that there's something special about the objects themselves, beyond just the process or results.

Summary

Overall, the three frameworks propose different ways of framing definitions of phenomena. One can think of situations as consisting of objects with particular properties, interactions between those objects, and the results of those interactions. If a definition only considers the results of the interactions, it is a *results-based* definition. If a definition considers how the objects interact (possibly in addition to results), then such a definition is a *process-based*

definition.[6] If the definition relies on properties that the objects have (possibly in addition to interactions or results of those interactions), then such a definition is a *property-based* definition.

As a clear example of the three frameworks, consider the term "being a good student." One might define "being a good student" as getting high grades. If all that matters is one's grades, then "being a good student" is defined under the Results Framework. Alternatively, one might think that scores aren't the only way to be a good student. Instead, a good student follows particular processes, such as studying for tests and learning the material. Even if they don't necessarily do well on the tests or get good grades, because of their behavior, they would still be considered to be a good student. Though, one might define it so that utterly abysmal grades can cause them to not be considered a good student, regardless of how hard they studied. If "being a good student" is defined in this way, as particular processes that one follows, even though results may play a role, then the definition is process-based. Finally, one might believe that "being a good student" is dependent on actually being a registered and enrolled student at a school. That is, someone who is not homeschooled and learns information online without any particular curriculum or required structure, studies the material, and scores well on exams they find and take, is not a student, merely because they are not registered as a student in any sort of school. Now, in order to be a good student, the person must not only be enrolled at a school, but also engage in studying and have decent results. Such a definition of "being a good student" is defined under the Properties Framework, because it relies on the properties of the person, namely, the property of being a registered student.

While the previous examples are fairly clear as to what framework they fall under, it is not always easy to place definitions under a single Framework. Here, I will describe a few of the

[6] The interactions between the objects form a *process*

common cases where it may be difficult to label a definition as either results-based, process-based, or property based.

Imagine a definition that has the form "X is defined by any process that results in Y." For example, "being a good student" might be defined as "following a process that results in good grades." In this case, even though it seems that the definition is focused on a process, it turns out that the only relevant portion of the process is the results. Thus, such a definition is *results-based*.

Next, imagine a definition in the form "an entity is X if and only if it behaves in a way that results in process Y being followed." In this case, it may seem that "process Y being followed" is a result, and therefore this definition should fall under the Results Framework. For example, one might define "being a good student" as "behaving in a way that results in studying" However, notice that the result itself is a particular process (studying) taking place. Thus, the definition does consider interactions, rather than just results, and so the definition is *process-based.*

Another case that may be confusing is definitions in the form "X is the case when result Y occurs, and result Y occurs when property Z is satisfied." For example, one might claim that a person is a good student when they score well on an exam, and scoring well on an exam is defined as the grade for that exam being the top grade. In this scenario, it may seem that the definition is property-based, since the grade may be considered to be a property of the exam. In cases such as these, we ask whether the property is the result of a relevant process. In this case, the grade on the exam is a result of the process of taking the exam. Thus, our definition is merely *results-based.*

Finally, imagine a definition in the form "X is the case when process Y is being followed, and only things with property Z can carry out process Y." For example, one might claim "a person is a good student when they are studying for a required exam." But, upon clarification, it turns out that exams are only considered to be required if one is a registered student. Then, since being a registered student is a property, and whether or not one can carry out the process is dependent on that property, such a definition is *property-based.*[7]

These three frameworks, which form the basis for results-based, process-based, and property-based definitions, will be used to structure our discussion of definitions of empathy, as they provide helpful ways of categorizing particular definitions and of diagnosing particular problems that may arise when using them.

Cognitive Empathy

Now with the three frameworks in place, we can begin to explore different classes of definitions of the term cognitive empathy. Generally, an entity is considered to exhibit cognitive empathy through determining how other entities feel, either in the current moment or in previous or potential future moments. However, what counts as "determining" will shift based on what Framework is used.

For results-based definitions of cognitive empathy, cognitive empathy is defined by the result of a process. That is, under the Results Framework, cognitive empathy is based solely on an entity's ability to determine what another entity's emotions are. Under this definition, it does not matter how an entity determines the emotions of another entity, as long as it does indeed determine the correct emotion. If we take counterfactual concerns into account, it may be that an

[7] One could argue that being a registered student is not a property of a person but rather the result of the process of "registering." If that is how one is considering the terms, then "being a good student" would be a process-based definition, but "registering" would then be a part of the process of "being a good student."

entity is considered to not have cognitive empathy based on how they *could* have failed to determine the correct emotion. Under counterfactually-sensitive results-based definitions of cognitive empathy, an entity that randomly guessed emotions and happened to be right would likely not be considered to have cognitive empathy. However, if the results-based definition is not counterfactually sensitive, then an entity that guessed emotions and happened to be right would indeed be considered to have cognitive empathy.

Under the Process Framework, cognitive empathy is based on following a particular process. This process is usually, although not necessarily, based on human cognitive empathy (Decety). Process-based definitions allow us to differentiate between different methods of having empathy, even if those two differing methods generate (and would always generate) the same results. For example, imagine that one entity watches a person's facial expressions to determine how they feel, while a second entity merely asks the first entity what their conclusion was about the person's feelings. Both entities will always end up with the same result, since the second entity merely copies the result of the first entity. Thus, under a results-based definition of empathy, both the first and second entities would have the same kind of cognitive empathy.[8] However, under a process-based definition of cognitive empathy, the two entities have very different kinds of cognitive empathy.

Under the Properties Framework, whether or not an entity has cognitive empathy is determined by properties of the entity. For example, one may believe that having a "mind" is required for cognitive empathy. In order to be a process-based definition, the definition of "mind" would need to require that an entity merely carrying out the processes of a mind or coming to the same computational results of a mind is not sufficient to determine that that entity

[8] Here, I am considering our results-based definitions to not be extremely counterfactually-sensitive. For example, one could imagine a scenario in which the first entity lies to the second entity. In that scenario, they would obtain different results. However, that seems like a fairly far-fetched scenario.

has a mind. For example, if one believes that humans have minds due to our biological basis of existence, but that machines without biological components cannot have minds, then even if a machine performed all the same processes as a human, including brain function, communication, self awareness, and bodily functioning, that machine would not have cognitive empathy. That definition of a "mind," which requires that minds are not defined by processes or results, combined with requiring minds to be essential for cognitive empathy, makes that definition of cognitive empathy a property-based definition of cognitive empathy.

Note that if an entity has cognitive empathy according to a property-based definition, then that entity must have cognitive empathy according to a process-based definition, but the reverse is not necessarily true. For example, if one believes that biological components are necessary for cognitive empathy, then if two entities, one with biological components, and one without, go through the same process to determine emotions, the entity with biological components may be considered to have cognitive empathy under both the process-based and property-based definitions, whereas the non-biological entity would only be considered to have cognitive empathy under the process-based definition.

Thus, the three Frameworks help to describe different classes of cognitive empathy that will be useful for future discussion.

Emotional Empathy

An entity is exhibiting emotional empathy when an entity feels the same emotion as another entity. People often engage in this sort of empathy when the emotion is sadness; when observing someone who appears to feel sad, it is natural to feel sad as well. Here, we tend to follow the definition given by Rogers: "The state of empathy, or being empathic, is to perceive the internal frame of reference of another with accuracy, and with the emotional components and

meanings which pertain thereto, as if one were the other person, but without ever losing the 'as if' condition. Thus it means to sense the hurt or the pleasure of another as he senses it, and to perceive the causes thereof as he perceives them, but without ever losing the recognition that it is as if I were hurt or pleased, etc. If this 'as if' quality is lost, then the state is one of identification."

While this definition successfully defines emotional empathy, a first reading of this particular definition may lead to assumptions which are not to be assumed in the following discussion. In particular, we will not assume that "to perceive the internal frame of reference of another" is always an action that is done consciously. This perception may be done without the entity being consciously aware that they have perceived the internal frame of reference of another. The idea that such perception may be possible without being consciously aware is similar to the "basic empathy" described by Fernandez and Zahavi. In addition, as is commonly done within this work, this definition is considered to extend to all entities, not merely humans, despite the occurrence of the term "person" within the definition.

From this definition, it is clear that having emotional empathy requires having emotions. Since emotional empathy requires feeling similar emotions to another entity, if an entity does not have the capacity to feel emotions, it cannot have emotional empathy.

By now, it is likely clear how the three Frameworks function, so the analysis here will be less involved. Under a results-based definition, having emotional empathy is determined by whether one is in fact feeling the same way as another entity. Under process-based definitions, an entity must go through a valid process of having similar feelings to another entity. And, under property-based definitions, emotional empathy requires going through a valid process with an entity that has the correct properties.

It is worth mentioning here that whether or not one's definition of emotional empathy is results-based versus process-based is contingent on how one believes emotions behave in general. If one believes that an emotion is a state of being, then one likely leans towards a results-based definition. However, if one believes that feeling an emotion is a process that requires time, then one likely leans towards a process-based definition of emotional empathy.

Summary

So, when can computers have empathy? If we assume that computers cannot yet feel emotions, then computers cannot have empathy. However, they may be able to have cognitive empathy. According to the most basic results-based definitions of cognitive empathy, a computer can have cognitive empathy merely by accurately determining what emotions another person has. According to more complex property-based definitions, computers must have minds and then must use those minds in the correct way to determine how other entities feel.

Let us now answer the three questions that originally motivated this section. If an entity randomly guesses how someone feels and happens to get it right, is that cognitive empathy? Under results-based definitions that are not counterfactually sensitive, yes. Otherwise, no. What if an entity is usually considered to have cognitive empathy, but now they are in a new situation and are often bad at determining how others feel — do they still have cognitive empathy? Under results-based definitions, the entity is considered to no longer have cognitive empathy. Under process-based and property-based definitions, the entity may still be considered to have a form of cognitive empathy, just a particular type of cognitive empathy that does not perform well in that scenario. What about if an entity appears to have emotions, but it doesn't have a biological body — can such an entity have emotional empathy? Under results-based and process-based

definitions, yes. Under property-based definitions that require a biological body for feeling emotions, no.

As we can see, the three frameworks help to define general responses to questions and categorize definitions of empathy. Overall, there are many different ways one can define empathy, but by at least determining which Framework one is using, one can generally determine a few key things about one's beliefs as to whether or not computers can have empathy.

Model Analysis

Now that we are clear on varying definitions of empathy, we can analyze whether our base machine learning model or any of its derivatives has empathy.

First of all, I think it is safe to assume that none of the models have emotions. They do not display any recognizable emotions, and there is nothing in the programming to suggest that they would, as far as I am aware. Thus, our models cannot have emotional empathy.

Immediately, it is clear that under property-based or process-based definitions that require human-like properties or processes, our model will fail to have empathy, because it does not follow human processes when it runs, nor does it have human properties.

However, results-based definitions may enable us to come to the conclusion that our model does have a form of empathy. Under a results based definition of cognitive empathy, our model has cognitive empathy if it is able to determine how another entity feels. As described in the related work section, determining how another entity feels is the goal of emotion recognition. Our model does something similar, but it doesn't detect emotions. Instead, it detects something *about* emotions: how quickly they change. How can we go about describing this sort of cognitive empathy? The next section covers metadativity, which describes exactly this distinction.

Metadativity

Imagine a type of semantic analysis program that, given a topic, determined a person's feelings towards that topic. This is in contrast to usual semantic analysis programs which, given text, attempt to determine the emotions expressed by that text. This imagined program would be much closer to the goal of my machine learning model, as one could imagine using such a program to continuously determine a person's feelings towards a topic, and then analyse the strength of the variation in their feelings over time. However, this level of analysis is quite different to the level that my program is attempting to accomplish. Similar to how a polarity detection program may not need to be able to detect particular emotions (unlike an emotion recognition program), a program which determines which topics a person is emotionally sensitive also does not need to be able to determine how a person feels. Instead, it only needs to be able to detect variation in emotions, not the emotions themselves.

In addition, if one is merely given topics that a person is emotionally sensitive to, it cannot be determined from that information what emotions, or what classes of emotions, the person feels about that particular topic. This is different than the relationship between polarity detection and emotion recognition, since if one has polarity information, this does give some information about what emotions a person is feeling. For example, if the polarity towards a topic is positive, and "anxiety" is considered to be a negative emotion, then, given the positive polarity, the person must not be feeling anxious about the topic. However if someone is "not sensitive" to a particular topic, one cannot deduce anything about how that person is feeling.

One might think that one may be able to deduce how a person is feeling from emotional sensitivity information by correlating emotional sensitivity with particular emotions. For example, one could look for particular psychological phenomena with emotional components

that are known to last for relatively short periods of time, such as the "fight or flight" response in humans (Controlling Angry People). Then, one might look for the emotions associated with such behavior, such as "anxiety" (Stefan). Then, it may be that a person being emotionally sensitive to a topic predicts that the person is more likely to have a fight or flight response to things within that topic. This would mean that one could predict that a person is more likely to feel "anxious" towards topics that they are emotionally sensitive to in comparison to other topics.

Even if such correlation is possible, however, it is never certain. Unlike polarity detection, if one knows that a person is not emotionally sensitive to a particular topic, it does not mean that they don't feel anxious. It may be that they are less likely to feel anxious towards that topic, but unlike a positive polarity, which rules out anxiety, information about emotional sensitivity does not rule out any emotions that a person could be feeling.

Thus, it seems that emotional sensitivity detection is in a new class of semantic analysis, which I shall call "metadative" semantic analysis. Semantic analysis is metadative when the output of the semantic analysis satisfies two criteria: 1. The output is not emotions (neither coarse or fine) and 2. The output gives no certain information about emotions. As described earlier, emotionally sensitive topic detection fits both of these bills, since topics are not emotions, and knowing the topics that a person is emotionally sensitive to does not reveal any certain information about how the person is feeling.

The term metadative was chosen because of its similarity to metadata. Metadata is data that provides information about other data. Traditionally, mobile carriers and operators would store metadata logs of who was using their phone to contact whom. This information was data about the messages, rather than the messages themselves. This sort of metadata has strong similarities to our definition of metadative semantic analysis. Information about who contacted

whom is not a message, just as topics are not emotions. In addition, while information about who contacted whom may be correlated with particular messages or classes of messages, it does not give any certain information about the content of the messages.

Analysis (cont.)

Thus, with the term metadative defined, it becomes clear that our machine learning system has a sort of cognitive empathy: metadative cognitive empathy. The system does not engage explicitly in cognitive empathy, but instead engages in predicting information *about* how others are feeling without predicting any explicit information about how others are feeling.

Even this description of our model, however, is not quite accurate. To demonstrate why this definition is inaccurate, I asked students from one of my courses to determine topics that *they* believed the authors were emotionally sensitive to. The results differed quite greatly from my dataset. So the model is not quite attempting to determine topics that the authors actually are emotionally sensitive to. Instead, the models are attempting to determine topics that I believe the authors are emotionally sensitive to. That is, the topics the authors are emotionally sensitive to *according to me*.

Accordativity

Sometimes entities may have particular types of empathy that don't reflect reality, but at least reflect how other entities perceive reality. If this is the case, we say that such entities have "accordative" empathy, because they have empathy "according to" some other entity.

For example, if two entities determine that a person feels sad because the person is crying, then both entities will have cognitive empathy (under a results-based definition) according to each other. If it turns out that the person is indeed sad, then the two entities will both have regular cognitive empathy as well. However, if the person is not sad, but instead is

crying because they are happy, then both entities will not have cognitive empathy, but they will both still perceive each other as having cognitive empathy, and thus they will still have cognitive empathy according to each other.

This terminology allows multiple entities to accurately describe the empathetic capabilities of another entity even if their own empathetic capabilities differ among each other. Instead of merely saying "this entity has emotional empathy," which could be understood as a very different emotional capability, depending on the audience, one can say "this entity has emotional empathy that is agreeable to me." Members of the audience can then say "the entity has emotional empathy that is agreeable to the speaker" without confusing one another as to what is being claimed. This terminology will be useful when discussing how we should say an entity has empathy, as well as during this analysis of the entity I have programmed.

Analysis (cont.)

Finally with metadativity and accordativity defined, we have a full analysis of the empathy of the machine learning models. The machine learning models that had perfect accuracy on the training set demonstrate results-based metadative cognitive empathy according to me for examples within the training set. Thus, while it can be said that many of our models exhibit a form of empathy, such a description is fairly complex.

Given that an accurate description is so complex, if someone were to ask me "Does your model have empathy?" how should I reply? While this section introduced tools for describing empathy, the next section determines guidelines for when we *should* say that an entity has empathy, and is the main conclusion of this thesis.

When should we say computers have empathy?

Throughout the course of this thesis, we have discussed many different topics regarding computational empathy. First, we discussed definitions of computers and definitions of empathy, using the three Frameworks to categorize differing definitions of empathy. We then used these definitions to describe how computers would need to behave under differing definitions of empathy. Next, we analyzed our machine learning models and developed tools for describing varying types of empathy that real computer systems may have. Throughout all of these discussions, the tone has been rather theoretical and general. Now, we move on to the big practical question: when *should* we say that computers have empathy? The main assumption in this portion of the thesis is that when we say a computer has empathy, *we should not be misleading.*

In order to achieve this aim, we will analyze strengths and weaknesses of definitions under each of the three frameworks in order to determine when particular definitions are more likely to be misleading.

Results-Based Definitions for Particular Scenarios

We have encountered many results-based definitions within this thesis, the most notable being a simple results-based definition of cognitive empathy, as it is under that definition that the machine learning models can be considered to have empathy. We discussed the implications of results-based definitions of emotional empathy, and how one might define "understanding" under the Results Framework, both with and without counterfactual considerations. We also discussed how "being a good student" could be defined under a results-based model if the results are grades.

There is one consistent theme throughout all of these results-based definitions: results-based definitions are good for *particular scenarios*. If I want to determine whether or not an entity has empathy *in a particular case*, we are usually looking at the *result* to determine the verdict. This is why my models can be considered to have results-based cognitive empathy within the training set - the training set is a set of particular scenarios.

In other words, when describing an entity's one-time performance, for example, by using the phrase "In this case, this entity is exhibiting empathy," a results-based definition of empathy is generally expected. It would seem odd to say "In this case, the entity failed to accurately determine the person's emotions, but the entity still exhibited empathy." Even if the definition of cognitive empathy is usually considered to be process-based, when we bring the definition into a particular scenario, we expect a results-based definition. To use a process-based definition or property-based definition would be misleading.

Property-Based Definitions for Counterfactual Robustness

Earlier on, we discussed how many descriptions of computers are misleading in odd ways from self-driving cars being described as "safe-drivers" to Siri being described as "intelligent." What is it that causes these definitions to be misleading? How can we avoid being misleading when labelling computers as "empathetic?"

My model, the self-driving cars, and Siri case all follow a similar pattern: within particular environments, the computer acts as we expect. However, when brought into new environments where usual functionality is still expected, the computer behaves radically different than our expectations. I would like to illustrate a few more examples where this is the case.

In the movie *I, Robot*, Detective Del Spooner is deceived in this sort of manner as well. In Detective Spooner's world, robots (which are a type of computer) live alongside humans

(Proyas). Detective Spooner initially believes that the robots in his world behave with human compassion. That is, he believes that the robots will use their human empathy capabilities to act like humans do. In fact, in almost all situations, this is indeed how the robots behave. However, he is shocked when he is in a car accident and watches as a robot makes the decision to save him over a young girl merely because he had a greater chance of survival. His model for the computers' empathy worked in almost all situations, until it broke down in a rare situation.

This particular type of deception regarding the capabilities of computer systems is particularly worrying especially since it may take place during times when humans are struggling. Before COVID-19, many large businesses ran computer models to determine how much product to buy and stock for consumers. But when COVID-19 was declared a pandemic, purchasing behaviors shifted rapidly, and the computer models were unable to accurately predict buying behaviors (Our Weird Behavior). Similar to Detective Spooner, it was when circumstances rapidly changed for the worse that the computer systems failed.

This concern is a concern about counterfactual robustness. That is, we have particular expectations of computer behaviors that function perfectly well. However, in situations that *could* occur, these expectations can be wildly wrong. Without counterfactual robustness, not only are we misled about the capabilities of the computers, we only find out that we are misled when things are not going well, precisely when we would likely rely the most on our computers to understand how we feel in order to help us.

For each of these cases with a counterfactual robustness concern, our model of how the computer will behave is based off of our model of how humans will behave, because the terms are terms that are generally applied to humans: "safe driver," "intelligence," "compassion," etc. The reason why our models of behavior did not work in new scenarios is because the entities we

were applying them to were electronic computers and not humans. They are not built the same way that we are, and therefore they do not perform the same way we would in all situations.

Even if an electronic computer was to follow the exact same process for empathy as a human, for example by following Pylyshyn's example of exactly simulating human behavior, because the computer is electronic, it will behave differently in particular circumstances than a human would (Cole). For example, if an electronic computer is used to simulate human cognitive empathy, such a system would likely not work in the rain. Simply because computers are currently made of different materials than humans, they must behave differently in particular situations. That is part of what it *means* to be made of a different material.

To the extent that humans are used to the computer's behavior in these counterfactual situations, defining terms under results-based or process-based definitions is not so worrying. However, it does not yet appear that humans are good at judging the failure cases of computers. No one expected the self-driving car to crash into the overturned truck (Stumpf).

Similarly, unless users are used to computer behavior in these counterfactual situations with respect to empathy, we should not use a results-based or process-based definition of empathy for computers. In other words, we should not say that computers have empathy, because doing so is likely to be misleading.

Conclusion

Throughout this thesis, we have analyzed multiple definitions for computational empathy and described cases in which computers may be considered to have empathy. We have created a model which exhibits a type of empathy described as results-based metadative cognitive empathy according to me. We have also analyzed general patterns in definitions of empathy and determined that results-based definitions are best used for describing computational empathy in

particular cases and that we should be wary of saying that computers have empathy as a general trait because people are not yet used to how computers will behave in varying circumstances.

My hope is that the information in this thesis can be used to more quickly determine better definitions of empathy for computers and describe empathetic systems more accurately so that people are more informed about the behavior of computers that are considered to have empathy.

Bibliography

Baumeister, Roy F., and Kathleen D. Vohs. *Encyclopedia of Social Psychology*. Sage Publications, 2011. *Open WorldCat*, http://www.credoreference.com/book/sagesocpsyc.

Blei, David M., et al. "Latent Dirichlet Allocation." *The Journal of Machine Learning Research*, vol. 3, no. null, Mar. 2003, pp. 993–1022.

Bojanowski, Piotr, et al. "Enriching Word Vectors with Subword Information." *ArXiv:1607.04606 [Cs]*, June 2017. *arXiv.org*, http://arxiv.org/abs/1607.04606.

Cambria, Erik, et al., editors. *A Practical Guide to Sentiment Analysis*. Springer International Publishing, 2017. *DOI.org (Crossref)*, doi:10.1007/978-3-319-55394-8.

Cole, David. "The Chinese Room Argument." *The Stanford Encyclopedia of Philosophy*, edited by Edward N. Zalta, Winter 2020, Metaphysics Research Lab, Stanford University, 2020. *Stanford Encyclopedia of Philosophy*, https://plato.stanford.edu/archives/win2020/entries/chinese-room/.

"Controlling Angry People." *Psychology Today*, http://www.psychologytoday.com/blog/let-their-words-do-the-talking/201101/controlling-angry-people. Accessed 26 Mar. 2021.

Decety, Jean, and Philip L. Jackson. "The Functional Architecture of Human Empathy." *Behavioral and Cognitive Neuroscience Reviews*, vol. 3, no. 2, June 2004, pp. 71–100. *DOI.org (Crossref)*, doi:10.1177/1534582304267187.

Foltz, Peter W. "Latent Semantic Analysis for Text-Based Research." *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, June 1996, pp. 197–202. *Springer Link*, doi:10.3758/BF03204765.

Hi, I'm Resistbot! https://resist.bot/. Accessed 26 Mar. 2021.

Kratzwald, Bernhard, et al. “Deep Learning for Affective Computing: Text-Based Emotion Recognition in Decision Support.” *Decision Support Systems*, vol. 115, Nov. 2018, pp. 24–35. *arXiv.org*, doi:10.1016/j.dss.2018.09.002.

Mikolov, Tomas, et al. “Efficient Estimation of Word Representations in Vector Space.” *ArXiv:1301.3781 [Cs]*, Sept. 2013. *arXiv.org*, http://arxiv.org/abs/1301.3781.

“Our Weird Behavior during the Pandemic Is Messing with AI Models.” *MIT Technology Review*, https://www.technologyreview.com/2020/05/11/1001563/covid-pandemic-broken-ai-machine-learning-amazon-retail-fraud-humans-in-the-loop/. Accessed 26 Mar. 2021.

Pedregosa, Fabian, et al. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research*, vol. 12, no. 85, 2011, pp. 2825–30.

Pennington, Jeffrey, et al. “Glove: Global Vectors for Word Representation.” *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014, pp. 1532–43. *DOI.org (Crossref)*, doi:10.3115/v1/D14-1162.

Picard, R. *Affective Computing*. MIT Media Laboratory, 1995, https://affect.media.mit.edu/pdfs/95.picard.pdf.

Proyas, Alex. *I, Robot*. 20th Century Fox, 2004.

Searle, John R. “Minds, Brains, and Programs.” *Behavioral and Brain Sciences*, vol. 3, no. 3, Sept. 1980, pp. 417–24. *DOI.org (Crossref)*, doi:10.1017/S0140525X00005756.

Stefan Bracha, H., et al. “Does ‘Fight or Flight’ Need Updating?” *Psychosomatics*, vol. 45, no. 5, Sept. 2004, pp. 448–49. *DOI.org (Crossref)*, doi:10.1176/appi.psy.45.5.448.

Stumpf, Rob. “Autopilot Blamed for Tesla’s Crash Into Overturned Truck.” *The Drive*, https://www.thedrive.com/news/33789/autopilot-blamed-for-teslas-crash-into-overturned-truck. Accessed 26 Mar. 2021.

“What Is TF-IDF?” *MonkeyLearn Blog*, 10 May 2019, https://monkeylearn.com/blog/what-is-tf-idf/.

Related Works

Decety, Jean, and William John Ickes. *The Social Neuroscience of Empathy*. MIT Press, 2009. *Open WorldCat*, http://site.ebrary.com/id/10282780.

Fernandez, Anthony Vincent, and Dan Zahavi. "Basic Empathy: Developing the Concept of Empathy from the Ground Up." *International Journal of Nursing Studies*, vol. 110, Oct. 2020, p. 103695. *DOI.org (Crossref)*, doi:10.1016/j.ijnurstu.2020.103695.

Nomura, Kohei, and Seiki Akai. "Empathy with Fictional Stories: Reconsideration of the Fantasy Scale of the Interpersonal Reactivity Index." *Psychological Reports*, vol. 110, no. 1, Feb. 2012, pp. 304–14. *DOI.org (Crossref)*, doi:10.2466/02.07.09.11.PR0.110.1.304-314.

Preston, Stephanie D., and Frans B. M. de Waal. "Empathy: Its Ultimate and Proximate Bases." *Behavioral and Brain Sciences*, vol. 25, no. 1, Feb. 2002, pp. 1–20. *DOI.org (Crossref)*, doi:10.1017/S0140525X02000018.