Artificial Intelligence (AI) is a transformative technology that mimics human intelligence in machines. AI enables machines to perform specific tasks that normally would take humans a significant amount of time. It has the potential to be used for a wide variety of services. It will revolutionize how we interact with the world and work. While building an AI application is extremely challenging and comprehensive, they will become more frequent in our lives. In healthcare AI aids in developing personalized treatment plans for diseases. In the automotive industry, it powers self-driving cars, enhancing safety and efficiency. AI algorithms improve customer service through chatbots and virtual assistants. In finance AI assists in fraud detection and black box algorithmic trading which is much more profitable then normal trading. AI also plays a crucial role in environmental conservation, predicting climate patterns and aiding in efficient resource allocation. These many applications showcase AI's potential to significantly impact and improve multiple aspects of our daily life and our workplaces. While AI can be used productively it also has the potential to be harmful from many different perspectives, a few of them being from a cybersecurity perspective, one example being polymorphic AI malware. For example the military uses AI with drones to recognize faces of enemies on the ground. AI is also being put into offensive and defensive computer systems. AI is intended to solve many things that humanity is lacking but yet we as a people don't fully understand how it works and how it comes to be what it is. A great analogy of this is the layering of the human brain. Imagine you were building a human brain layer by layer with glue but you had to let it dry and see if each layer came out right before continuing. It would be tempting if one layer came out wrong to just glue over it. This is what building AI feels like. There is no set programming; it's extremely comprehensive and confusing. The system acts in ways you would never expect it to at times. It says random things. It's frustrating but rewarding at the same time because the final product is so smart and advanced that it is worth all the time put into it.

Building an AI application from scratch is an extremely long process and takes a significant amount of time and effort for a developer team; a normal AI team is around ten people. Building an AI solo can take around a year. Making it give you good responses to user input all the time can take even longer. There is a certain way of thinking that you have to indoctrinate when starting to build a AI this type of thinking in layman's terms would be described as a computational systematic way of thinking what this means is you are thinking of everything that you code as a flowchart with many different outcomes and entrances. It's a very complex way of thinking and takes awhile to get used to. Once you start thinking in this manner you can begin to plan out the series of steps and processes needed to start to build an AI from scratch. Some key terms and definitions that are necessary to know about AI before everything.[1]

Algorithm: A set of rules or instructions given to an AI system to help it learn from data usually math based.

Recurrent Neural Network (RNN): A type of neural network that is well-suited to sequence prediction problems because it can maintain information in memory over time.

[1] I drafted my own definitions and used Google's BERT summarization tool to refine the definitions.

Loss Value/Function: A value that is spit out that is for evaluating how well a model is responding to the given data and how it is training on that data.

Overfitting: A modeling error which occurs when a model is too fit to a set of data points and fails to generalize to the data to generate diverse responses resulting in varied loss values between higher and lower during training.

Underfitting: Is when a model is too simple both in terms of the architecture or the data it is being given.

Decay Steps: Decay steps refer to the rate at which the learning rate decreases over time during the training process.

Transformers: A type of neural network architecture that is mostly used in python that relies solely on attention mechanisms to draw dependencies between input and output effective in natural language processing (NLP).

Sequence-to-Sequence (Seq2Seq): A model that processes a sequence of inputs like a sentence to generate a output commonly used for chatbots and dialogue systems

Attention Mechanism: A technique in neural network architectures that enables the model to focus on different parts of the input sequence to understand conversational context very important where the context of a conversation is crucial.

Self-Attention: A mechanism within an attention architecture that allows models to weigh the importance of different words within the same input crucial for understanding context and relationships in text also sometimes used for AI to understand emotion.

Tokenization: The process of converting text into smaller pieces like words or subwords which can be processed by models or removing pieces of text.

Embedding Layer: A trainable layer in neural networks that is used to convert categorical data especially words into numerical vectors that capture into something that has meaning to the computer/program.

Beam Search: A heuristic search algorithm that explores a graph by expanding the most promising node/choice/word in a limited set of options/data

Top-K: A probabilistic technique used in language models where the next word is chosen from the K. K being the most likely option according to the model's predictions. Top K helps introduce randomness into the text generation process making it potentially more diverse in its outputs.

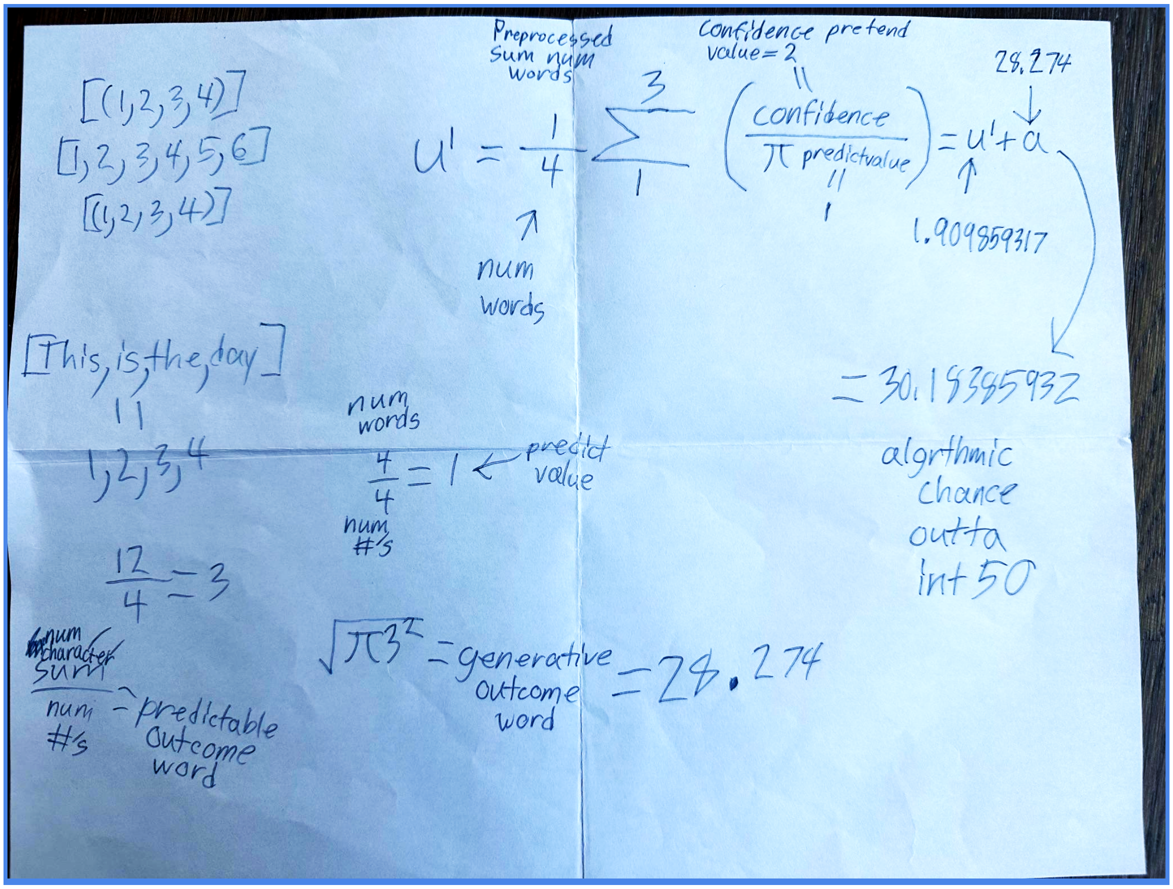
Hyperparameters: Parameters that set the training process of machine learning models/AIs to improve their performance on unseen data.

Fine-tuning: A technique where a pre-trained or untrained model is adapted for a task by continuing the training process with a dataset targeted to that task.

The first step an AI goes through is called Data Collection. This is a pretty self explanatory step; it involves sifting through different sources of information and picking out data that is useful for training. There are many different sources and types of information that can be utilized. Some of the file types that can be utilized to train AI are JSON, CSV, RTF and TXT files. For my project I chose to use TXT files as this seemed to be the most straightforward, and I was most familiar with this. In regards to collecting data to be used for training purposes I went to several different areas I would read through health journals, therapy handbooks and custom copyrighted datasets I gained access to. I would not directly copy any of the data from these sources, but I would create pairs of theoretical inputs and outputs that the AI could train off of to learn basic human conversational interactions. The data was classified into 6 intent based categories that are not labeled in the program or data: greetings, conversational connectors, therapy responses, emotional responses, ending statements/goodbyes and lastly misc responses/non classifiable conversational responses. These 6 intent based categories I created to organize the data when I was creating it were extremely useful in making sure that there was enough information available for the AI to train and create diverse and correct responses from the data.

I created precisely 598 data points which equates to the amount of lines in the TXT files. 598 data points is a relatively small dataset in the context of other AIs (A normal dataset is somewhere in the range of millions of data points!). The data has to go through a process called data cleaning which is where you sift through the inputs and corresponding outputs line by line. With data cleaning you make sure there are not a series of things in the data: profanity, threats to harm someone or something, computational errors, statements or explanations violating the normal acceptable societal standards of conversation and any ethical and moral issues. All of these things specified can not be in the data and must be cleaned out. If not you will get a very bad AI system once it is fully trained. Now there will always be some things that slip through the cracks like profanity here and there but the goal of data cleaning is to get the worst of the bad content out of the data. Another objective of data cleaning is to fix any missing values within the data, for instance spelling mistakes or numerical errors and inconsistencies within the data. Data cleaning is an essential part of the data collection process; it allows for the data to become more fine tuned and finalized for the AI to become trained on it.

For every AI there is an algorithm that is chosen for it and the algorithm suits the AIs needs and requirements. For my AI program which is a mental health chatbot I created my own custom algorithm for generating responses while also utilizing tensorflow LSTM models which are open source and available for the public to utilize as a python library. My program is a combination of many different types of Algorithms. I utilized top k, temperature and reranking of response candidates to enhance the responses correctness and coherence while also using these functions to improve upon the generative capabilities of the program. My program also included an attention model so that the AI somewhat understands the path and pattern the conversation between the user and the bot is taking. My own algorithm which I made and will provide below is a combination of many different kinds of algorithms.


[(1,2,3,4)]
[1,2,3,4,5,6]
[(1,2,3,4)]
Preprocessed sum num words
Confidence pretend value=2
28.274
$u' = \frac{1}{4} \sum_{1}^{3} \left(\frac{\text{confidence}}{\pi \text{ predictvalue}} \right) = u' + a$
1
num words
1.909859317
= 30.18385932
[This, is, the, day]
1,2,3,4
num words
$\frac{4}{4} = 1$
predict value
num #'s
algrthmic chance outta int 50
$\frac{12}{4} = 3$
~~num~~ character sum
num #'s
= predictable outcome word
$\sqrt{\pi 3^2}$ = generative outcome word = 28.274

These types of algorithms like the one I made are used to calculate the probability of the next word in the sequence of generation of a response. For example, If I train the AI with a few sequences of data where there is an input that goes “Hello how are you” and the output pair associated with that is “I'm doing well how are you doing today” then there is another pair in the data where the input is just “Hello” and the output corresponding to the pair is “Whats up how are you”. Then once the AI is fully trained from these example sequences in theory if I input “Hello” the AI will run through its generation algorithm and might say as an output “Hello whats up How are you doing today”. As you can see the last 2 words were generated by the generative algorithm and the first 3 words were reorganized to make a more creative response from the AI. When the algorithm is generating a more creative output response, temperature and top k have a significant amount of effect in this action. So by this example it can be seen that the output was generated from the algorithm based on the training data because it noticed a similarity in the data and wanted to expand on it.

Even before the generative algorithm is utilized to generate responses from the AI, the AI must go through a step called Preprocessing. This is a step that utilizes the data that is provided for training and removes all unwanted characters and punctuation that is not needed. The Preprocessing also puts tags at the beginning and end of the sentence so that the program recognizes the sentence it puts <start> at the beginning and <end> at the ending of the sentence; this is one of the last steps. Preprocessing allows for the data to be cleaned before it is inputted into the AI for training whether that be within the feedback model or the regular training process it depends. Preprocessing is meant to make the data available, cleaned and suitable for the specific requirements of the task that is trying to be achieved based on the user input. Preprocessing is crucial to the development of an AI system because without it you would have raw unfiltered data that would have errors and mistakes within it. The performance and accuracy of model responses depends on the cleanliness and amount of inaccuracies in the data. A great example of how Preprocessing works is:

Preprocessing steps
Tokenization "I do not like green eggs and ham" ["I", "do", "not", "like", etc...]

Stop word removal get rid of common words like "the", "or", "and", "a"

Stemming/Lemmatization Shorten words to their root by removing endings like "ing", "ed", "ion" and so on

Punctuation removal “hello how are you?” turns into “hello how are you”

```
from sklearn.naive_bayes import MultinomialNB
nb_model = MultinomialNB()
nb_model.fit(X_train, Y_train)
Y_pred = nb_model.predict(Y_test)
accuracy = accuracy_score(X_test, Y_pred)
```

This example piece of code can be used to predict a new class of new import data

This example is a series of basic notes regarding how Preprocessing works. The example showcases tokenization and punctuation removal among other types of preprocessing utilized. It shows the removal of certain characteristics in the data to prevent overfitting and something I call overpatternization of the dataset. The sample also shows an example of code for preprocessing from an AI class I took last summer.

Once the data is fully preprocessed and all the extraneous characters are removed from the data and cleaned, I enter all the data available. Using two TXT files, inputs.txt and outputs.txt, I then separate the data into trainable text in which the AI will train off the theoretical user input data and the corresponding expected theoretical outputs of an AI. The data available will heavily influence the generative capabilities of an AI without a large dataset, the AIs responses will be dissatisfactory. I decided with my program not to use a validation dataset as I thought it would overcomplicate things. This is not the normal approach when building an AI. I also did not utilize cross validation as this seemed excessive.

When accessing an AIs performance many things are taken into account: accuracy of responses, confidence, response time, rewards for responses, sentiment analysis and intent classification. These performances indicate how well an AI is performing in response to the responses it is giving out. I decided when building my AI system that it would be suitable to have a log file, the reason for this being that it would make it much easier for me to see the ins and outs of how the computer and AI were processing the information being fed to it. The way I would see this is by the hyperparameters that the log file would display. A example of a display on the log file would be this:

```
absl - INFO - SubwordTextEncoder build: trying min_token_count 328
absl - INFO - SubwordTextEncoder build: trying min_token_count 164
absl - INFO - SubwordTextEncoder build: trying min_token_count 82
absl - INFO - SubwordTextEncoder build: trying min_token_count 41
absl - INFO - SubwordTextEncoder build: trying min_token_count 20
absl - INFO - SubwordTextEncoder build: trying min_token_count 10
absl - INFO - SubwordTextEncoder build: trying min_token_count 5
absl - INFO - SubwordTextEncoder build: trying min_token_count 2
absl - INFO - SubwordTextEncoder build: trying min_token_count 1
root - INFO - 2024-03-08 23:11:22, User Input: Hello, Bot Response: Im always here to provide answers What can I assist you with, Response Time: 9.563585, Reward: 1
root - INFO - 2024-03-08 23:11:40, User Input: Nothing much Goodbye, Bot Response: Im here if you need me for anything else, Response Time: 6.651177, Reward: 1
absl - INFO - SubwordTextEncoder build: trying min_token_count 328
absl - INFO - SubwordTextEncoder build: trying min_token_count 164
absl - INFO - SubwordTextEncoder build: trying min_token_count 82
absl - INFO - SubwordTextEncoder build: trying min_token_count 41
absl - INFO - SubwordTextEncoder build: trying min_token_count 20
absl - INFO - SubwordTextEncoder build: trying min_token_count 10
absl - INFO - SubwordTextEncoder build: trying min_token_count 5
absl - INFO - SubwordTextEncoder build: trying min_token_count 2
absl - INFO - SubwordTextEncoder build: trying min_token_count 1
```

What this example from the log file tells me is that the program is building the tokens correctly. The file also tells me whether or not the program is giving out a good response or not. It provides me with the amount of time it took for a response to be generated which can be useful to determine if there are any issues with the response generation of the program. The log file also includes the reward system and whether or not the response that the program gave out will get a reward or not. These hyperparameters in the log file along with the other parameters that are hard coded into the program like decay steps and the decay rate have a major impact on the quality of responses the AI will give out. The hyperparameters also are required to be adjusted sometimes if the AI is not giving the result that I want or that is needed.

One of the last steps an AI goes through is called post processing; this is a very important final step. What post processing does is it refines the raw output that an AI spits out. It is a series of operations or techniques that get applied to the raw output to make the results of a AIs responses more suitable to its needs and requirements. I made my post processing a little different than you normally would with an AI I had a specific design in mind. My post processing method operates like this: the method takes the generated token output by the model and turns the tokens into human readable text using the tokenizers decode section of the method. The post processing then removes the <start> and <end> tags from the sentence these tags were added during preprocessing; these tags get replaced with an empty string. This erases the tags from the output the user would see. Then lastly the post processing takes any trailing whitespace after the removal of the tags and removes the extra trailing whitespace this ensures that the output looks nice and clean for the user to see.

Something that is very important after post processing is ensuring that the AI is giving good responses back to the user based on their input. This was a major part of my program and helped it become what it is. The way I achieved improving the responses was extremely challenging. I used a combination of things: a feedback system, reward system, confidence of response correctness and a relevance/sentiment checker. So for the feedback system I utilized reinforcement learning from human feedback (RLHF). This RLHF is a reinforcement learning system that incorporates human evaluation into its training. The goal of RLHF is to improve the model actions/performance based on the reward signals/response corrections from the human evaluator. The way I did this was the AI generated a response the response then gets evaluated for relevance/sentiment and confidence it gets given a confidence and relevance score. If the confidence score is below the threshold or the response is deemed not relevant by an external model then the AIs response gets prompted for human review. What this means is that the user gets given a prompt to correct the response of the AI. These corrected responses can then be used to retrain and refine the AI and help it better understand the complexities of human interaction and conversation.

For the reward system I created it so it uses the relevance/sentiment checker. The system assesses the sentiment and relevance based on the user input and then the corresponding AIs response and assigns a reward. The way the reward assigning works is if both the sentiments align and the response is relevant to the users input then a positive reward is assigned. Otherwise if the sentiments don't align or the response that is generated is irrelevant to the users input then a negative reward is assigned. What this reward system achieves is it makes the AI want to generate responses that are relevant and are in line with the human emotion that the user input is portraying.

Collected feedback, the interactions that get flagged for retraining due to low confidence or irrelevance and the corrected responses are then used for the AIs retraining to improve response performance. The many systems I have created for my AI are very useful in facilitating growth and learning performance over time. The systems help in enhancing responses and also allowing for human intervention when necessary in the constant reinforcement learning process. These many systems increase reliability and relevance of users to AI conversations and interactions thus ensuring responses become better and more correct over time with training.

The last part of building an AI is the monitoring of the program's performance overall, some of these indicators being the responses, confidence levels and systems that I built to maintain and monitor performance that were mentioned before in previous paragraphs. Maintaining and making sure everything is working properly is one of the most important parts of building an AI. Tracking these indicators not only helps in ensuring that the AI operates properly and within its intended parameters but also identifies any issues that are arising suddenly or slowly. Maintenance and updates based on indicators/hyperparameters also help continually enhance and improve the AI and its performance.

Building this AI application has encompassed a series of steps from data collection to monitoring and maintenance. This project demanded so much technical expertise but also an understanding of the ethical limitations of building an AI. The development process was extremely complex and designing a system like this was extremely difficult because I taught myself everything I know. AI is going to revolutionize the world we live in today. Around 60% of the workforce will be replaced. AI will increase efficiency and offer solutions to problems we didn't believe as a human race were possible.

Building an AI application, specifically a mental health chatbot as I did shows the potential for Artificial Intelligence to contribute positively to society by offering people services which provide support and assistance where it wasn't before. However when I was building the system it became extremely obvious how important it is for human oversight for the training and feedback while developing these products.

While AI becomes consistently more embedded into our everyday lives the need to be able to understand how it operates and works is essential. The process I have outlined about how I have built my AI system could be helpful to people in the future trying to develop their own and foster their knowledge on the topic of AI. My hope is in the future the world will be able to mitigate the risks AI poses according to them and use AI to benefit and correct issues in the world and bring about positive change using the new technology.

Works Cited

"Amdahl's Law." *Wikipedia*, Wikimedia Foundation, 26 Mar. 2024, en.wikipedia.org/wiki/Amdahl%27s_law.

Arxiv.org. arxiv.org/pdf/2307.06435.pdf.

Chatterbot-corpus. GuntherCox, github.com/gunthercox/chatterbot-corpus/tree/master/chatterbot_corpus/data.

Microsoft Research. Microsoft, www.microsoft.com/en-us/research/publication/adversarial-training-for-large-neural-language-models/.

Quora. "First-Quora-Dataset-Release-Question-Pairs." *Quora*, quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Small-english-smalltalk-corpus. Zeloru, github.com/zeloru/small-english-smalltalk-corpus.

Williams, Adina, et al. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference." *Aclanthology.org*, aclanthology.org/N18-1101.pdf.

Code for my program:
https://github.com/BJW333/JOSH