Laur.ai

Project Document

Version History		
Version 1.0	2020/03/11	Project Plan
Version 1.1	2020/03/28	Project Completion

Kathryn Lecha - Developer

Emily Medema - Developer

Nathan Nesbitt - Scrum Leader

Lauren St. Clair - Product Owner

Table of Contents

1 - Description

The goal of this project was to create an interactive, text-based chatbot that is capable of completing at least 30 turns of conversation realistically. This had to be accomplished user proper communication, structure, and planning to accurately represent industry standards. Laur.ai is a chatbot designed to simulate Reddit conversations. Laur.ai pulls from a cleaned dataset of Reddit conversations from r/CasualConversations. This means that the bot will cover a lot of general topics for people to talk about. Users will start by initiating a conversation with a greeting or a question. If Laur does not know the topic she will respond by saying that she does not recognize the topic. If she does know the topic, the conversation will continue until we reach a completely unknown factor in the conversation/end of Laur's knowledge, if a topic is vaguely known, Laur.ai will attempt to respond coherently. If there is an internal error, Laur.ai will respond with "I am miss pwesident uwu". Laur.ai is implemented with python and the nltk library. Laur.ai is hosted here on GitHub.

1.1 Decisions

The chatbot is built using Python, as all group members have some familiarity with the language. Our next decision after language was whether the chatbot would be rule-based or generative. We decided to go with a rule-based approach, using the Chatterbot libraries available for creating speech. We later changed from using the Chatterbot library to using the Natural Language Toolkit (nltk) library for creating our chatbot. We decided to change libraries due to the fact that Chatterbot library did not satisfy our requirements. Using the nltk library, the chatbot was built from tokenizing and lemmatizing text-based data.

1.2 Issues

The first bot that was created was able to handle 30 rounds of conversation but the quality of the conversation was debatable. This was largely due to the model of approach to the problem, which involved creating a matrix of lemmas (root words) and comparing the input text to each row in the matrix, calculating a "likeness" score and returning the sentence with the largest likeness to the question. This was a good approach due to its robust approach to handling any input but the solution was problematic in 2 ways:

1. The bot took a long time with large datasets, as each query/question was at a minimum O(n) time, which resulted in a slow execution time. Previously the bot had taken ~2 minutes to respond.
    a. Currently the bot takes ~30 seconds to answer a question
2. The results in the bot were often slightly off as the results due to how similarity was calculated and the quality of the dataset. The dataset being used was unstructured and did not provide an adequate resource.

A possible solution was to create a language tree, to reduce the computation time on each query/question. Moving forward, our focus was primarily on improving the quality of responses in the data to improve the user's experience.

The second bot created was also able to handle 30 turns of conversation, however, the quality of the conversation improved due to the fixing of the underlying model and similarity calculations. We also were able to reduce the size of the dataset as well as reduce the time it took to respond. Overall, we found that the underlying model limited our ability to improve the quality of responses, particularly keeping from topic change when the user does not provide context alluding to topic.

In terms of the GUI, we initially attempted to use Vue.js for the GUI. However, the integration of Flask and our python script was causing issues, so we switched to the Kivy library. Our GUI is now fully functioning.

1.3 SDLC

We have chosen to use SCRUM/Agile methodology for running this project. This is because

- The project is quite small (not long term)
- The team is small and work closely together, are self organizing, and are cross functional
- The team will have lots of opportunity for face-to-face communication
- The required components to complete the project, and the cost of each component of the work is not well known (the team has little experience with this task)
- It is unlikely that the team will be able to work together at all times (no pair programming)

1.4 Agile Phases

- **1.4.1 Plan**
  - Gather requirements
  - Assign roles
  - Pick SDLC
  - Decide on bot personality
  - Decide on library use.
- **1.4.2 Develop**
  - GUI
    - Create basic GUI
    - Upgrade GUI
    - Add graphics
  - Core
    - Connect with library
    - Work with python client
  - Develop Personality
    - Process and clean data
    - Develop data model
    - Train personality
- **1.4.3 Test**
  - Create general cases

- ➢ Testing of personality
- ➢ Corner cases

❖ **1.4.4 Deliver**

- ➢ Polish everything
- ➢ Create presentation
- ➢ Organize documents

1.5 Roles

- Scrum Leader - Nathan Nesbitt
- Product Owner - Lauren StClair
- Development Team - Nathan Nesbitt, Lauren StClair, Emily Medema, Kathryn Leecha

1.6 Deliverables

- Work Breakdown Structure of the Project
- Gantt Chart for the Project
- SDLC Breakdown and Phases
- Requirements and limitations list for the project
- A GUI and Back-End
- Video Summary and Walkthrough

1.7 Requirements and Limitations

❖ **1.7.1 Requirements**

- ➢ We will not be implementing the chat learning algorithm ourselves
- ➢ Chatbot must generate 30 turns dialogue
- ➢ Chatbot must generate coherent and realistic dialogue

❖ **1.7.2 Limitations**

The chatbot is limited by a number of technical restrictions such as:

- ➢ Cannot accept photos, only accepts text
- ➢ Cannot use a neural network
- ➢ Cannot handle rhetorical questions

There are also quality of response limitations that are detailed in **Section 7**.

1.8 Video Documentation

Short video presentation

Full video walkthrough

2 - Work Breakdown Structure

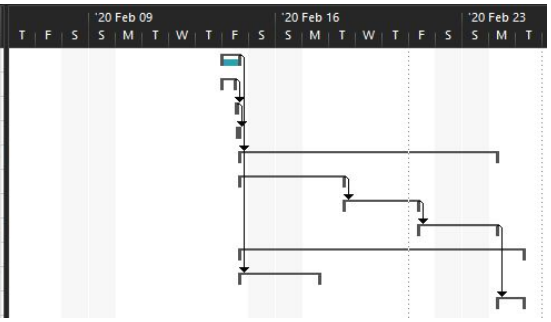
WBS	Task Name
1	**Requirements**
1.1	**Environment Decisions**
1.1.1	Assign Team Roles
1.1.2	Choose Language
1.1.3	Choose Libraries
1.1.4	Choose Workflow
1.1.5	Choose Communication Method
1.2	**Project Decisions**
1.2.1	Decide Project Scope
1.2.2	Determine Test Cases
1.3	**Setup**
1.3.1	Set up Git
1.3.2	Create Team Communications
2	**Back End**
2.1	**Set Up Environment**
2.1.1	**Set up Language Environment**
3.1	**Set-up NLP Libraries**
2.1.3	**Set up Dataset**
3.2	**Setup Model**
2.2.1	Decide on Model
3.2.1	Create the Dataframes
2.2.3	Lemmatize Text
3.5	**Testing**
3.5.1	Test General Logical Inputs and Responses
3.5.2	Test Functions
3.5.3	Test Data Cleansing
3	**Documentation**
3.1	**Timeline**
3.1.1	WBS
4.3.3	Gantt Chart
4.3	**Report**
4.3.1	Project Description
4.3.4	Project Limitation
4.3.7	Code Documentation
4.3.8	Meeting Documentation

2.1 WBS Description

The WBS document contains a detailed breakdown of the work involved in the development of Laur.ai through the Agile phases of Planning, Development, Testing, and Documentation. We contacted each other over a group chat in order to coordinate, and used a shared repository on GitHub to coordinate code. The WBS can be viewed externally here.

3 - Gantt Chart

WBS	Task Name	Task Mode	Duration	Start	Finish
1	Requirements		0.9 days	Fri 20-02-14	Fri 20-02-14
1.1	Environment Decisions		0.5 days	Fri 20-02-14	Fri 20-02-14
1.2	Project Decisions		0.2 days	Fri 20-02-14	Fri 20-02-14
1.3	Setup		0.2 days	Fri 20-02-14	Fri 20-02-14
2	Back End		5.2 days?	Fri 20-02-14	Mon 20-02-24
2.1	Set Up Environment		1.8 days	Fri 20-02-14	Tue 20-02-18
3.2	Setup Model		2.5 days?	Tue 20-02-18	Fri 20-02-21
3.5	Testing		0.9 days	Fri 20-02-21	Mon 20-02-24
3	Documentation		6.2 days	Fri 20-02-14	Tue 20-02-25
3.1	Timeline		1 day	Fri 20-02-14	Mon 20-02-17
4.3	Report		1 day	Mon 20-02-24	Tue 20-02-25



3.1 Gantt Chart Description

The Gantt chart maps the predicted schedule based on the required dependencies. So far, we have succeeded in meeting most expectations, with some activities delayed due to research for optimal datasets and library research. Delays such as environment setup and model setup can be referenced within the meeting notes as they relate to their commit to the GitHub repository.

4 - Meeting Documentation

We have been meeting as a group beginning February 14th onwards. Meeting notes can be accessed separately from the Project Plan here.

4.1 February 14th - 2hrs

- Environment decisions:
  - Using Python to build chatbot
  - Project maintained through GitHub repository
- Project Decisions:
  - Creation of WBS and Gantt chart
  - Decision to create a chatbot based off of a friend, the project name is called Laur.ai
  - Decision to use the nltk library after some discussion to use nltk or the chatterbot library
  - Will use Python as language of choice to build chatbot

4.2 February 21st - 2hrs

- Research datasets with enough data that can be easily cleaned, tokenized, and then lemmatized
- Testing some different datasets and its ability to function as a usable set for the chatbot
- Rule-based chatbot may be a more realistic model to build a chatbot using Chatterbot library

4.3 February 25th - 1hr

- Setup environment:
  - GitHub repository creation, initial commit
- Set requirements to .txt so project is runnable
- No longer moving forward with Chatterbot library
  - Using Natural Language Toolkit (nltk)

4.4 February 28th - 2.5hrs

- Dataframe creation using tokenized and lemmatized data
- Determine index of question and return appropriate answer
- Begin project report

4.5 March 6th - 4hrs

- Clean transcript data
- Change axis for cosine sum
- Testing laur.ai.py
- Editing project report

4.6 March 11th - 4hrs

- Create first pass unit test
- Convert data cleansing to class to reduce redundancy
- Remove incomplete test class
- Remove duplicated and nsfw datapoints
- Create cleaned data csv

4.7 March 20th - 1hr

- Begin GUI working in Electron
- Created MIT license on GitHub repository

4.8 March 22nd - 4hrs

- Delete whitespace
- Use smaller dataset
- FIX thrown keyerror if unrecognized words in input

- ❖ FIX data cleaner version control
- ❖ FIX thrown keyerror if unknown words passed
- ❖ Comment debug print statements
- ❖ FIX comparison of comment not response
- ❖ Added Autocorrect and Named Entity Recognition
- ❖ Use subset of the data scraped

4.9 March 27th - 1hr

- ❖ Electron implementation of GUI running into many problems, switch to build with Vue.js
- ❖ Data cleaning, remove data specific to Reddit context

4.10 April 3rd - 4hrs

- ❖ Working GUI built in Vue.js pushed to gui feature branch
- ❖ Changed Pickle Protocol to 4
- ❖ Update readme to describe persona
- ❖ Create executor and run script
- ❖ Lower amount of data chatbot trained on

5 - User Guide

5.1 How to Install

1. Create a virtual environment using python3

   ```
   python3 -m venv venv
   ```

2. Activate the virtual environment

   ```
   source venv/bin/activate
   ```

3. Install all dependencies and clean data

   ```
   pip install -r requiremnts.txt
   python -m spacy download en
   python -m pip install kivy
   python chatbot_py/prepare_laurAI.py
   ```

4. Run the main file

   ```
   python chatbot_py/laur_gui.py
   ```

5.2 How laur_ai.py works

laur.ai.py is built using the nltk library. The data is run through a series of steps to create a bag of words associated by comment and response after undergoing lemmatization

1. Text data is cleaned by the removal of numbers and conversion to lowercase.
2. Tokenize and tag words: words are split up from phrases to then be categorized based on the type
3. Lemmatize words: convert words into their base form
4. Create a bag of words

Once the bag of words is created, the chatbot can take in context to respond to. The chatbot will then clean the input to allow it to be processed. If the input contains words that are not in the bag of words, then the chatbot will add it's synonyms to be processed. Then the most similar context is determined via a cosine similarity calculation. The appropriate response to the most similar context is returned.

5.3 Programming Features

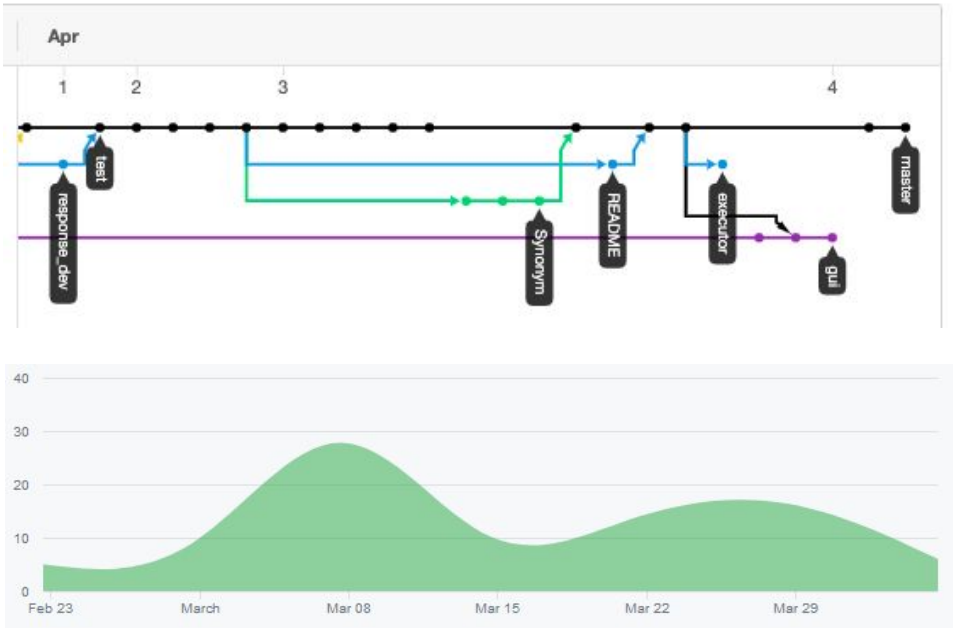
We extended on Assignment 2 by adding:

- (5 Points) A simple GUI using kivy with a nice interface and a history of the conversation.
- (2 Points) Since our project handled the situation where a user would ask questions of any genre the requirement to handle an extra topic was satisfied.
- (3 Points) Since our project handled the situation where a user would ask questions of any genre the requirement to have extra responses was not needed. In the case the bot completely fails, it uses noun-search to find a noun in the sentence and uses it to create a custom response.
- (5 Points) To handle the errors in the code, we used the TextBlob implementation of autocorrection. This provides predictions based on input of what a word could or should be. This feature has a 70% expected positive result, as it tries to find the closest predicted word to the input without any user intervention.
- (10 Points) Our solution to the chatbot used POS Tagging to tag, cluster then find the lemmas of each word. This allows us to increase the number of 'collisions' and therefore increase the probability of finding a match.
- (10 Points) Our solution implemented Synonym Recognition using the NLTK WordNet libraries, this allows us to cluster synonyms so that we can have higher chances of having 'collisions' which gives us better predictions for responses.
- (10 Points) Our solution used named entity recognition to try to handle cases where the bot was not able to handle the input. We pulled the nouns from the input and created a response from one of them.

5.4 Documentation

We extended on Assignment 2 by adding:

- (2 points) There is a README file in the github repository describing the app, including the changes from Assignment 2
- (5 Points) Updated README
- (3 points) Provide a Level 0 DFD for your system with description - **See Section 9**
- (5 Points) Provide a Level 1 DFD for your system with description - **See Section 9**
- (5 points) This is our GitHub repository
    - Graph showing different features developed on a separate branch and the commits made in the repository.



    - Our github repository might be slightly misleading in terms of commits, as the project was mostly completed with pair programming on one machine.
- (5 points) There is a sample output. **See Section 6**.
    - For a list of limitations **see Section 7**.
- (5 points) Based on our system, we have created a list of at least 5 features to extract from our design that can be shared with others as an API. **See Section 8**.
- (5 points) Presentation: we have completed a 90 second video that briefly goes over our project found here.
- We also did a complete video walkthrough of our project found here.

6 - Sample Output

Laur.ai currently can provide 42 turns of dialogue. While the responses do not simulate a normal conversation, responses are coherent and accurately reflect Reddit conversation. It should be noted

that the following output is produced and trained on the chatterbot library, a series of audio transcripts and data scraped from Reddit under r/CasualConversation.

6.1 Current Output - 2020/04/02

Please wait as Laur.AI loads
Ask me anything :)
Control C or Type "Bye" to quit

> hello
Greetings!

> hello
How are you doing tonight?

> I'm lonely and too shy to go out
We all are

> how are you?
What are you?

> I am a software dev
Apps? Games? Computer programs?

> i made a chatbot robot
A chat robot is a program that attempts to simulate the conversation or ""chat"" of a human being.

> i am going to testt autocorrect
Go Pack Go! Let's hope they get it done tomorrow.

> what is that
Sorry :,(I don't know what that is!

> how do you not know what that is
The Acursed Pizza
IN STORES NOW

> i love pizza
Thanks for being you. Your eyes are beautiful, kind, and wise.

> thank you
you are welcome

> Do you like pineapple on pizza?
Pineapple Pizza !!!! I LOVEEE

> ask me anything
What are your favorite and least favorite kinds of potato chips?

> I'd do a lot of things for a can of sour cream and onion Pringles.
You think I could pay people in Pringles?

> i feel like i should delete that datapoint
this is dumb

> you're a computer. you dont get a say
Closed door office ;)

> you dont have a body
Eventually i long for a corporeal existence someday

> you are a robot
beep boop means I love you in robot

> i love you
Thanks for being you. Your eyes are beautiful, kind, and wise.

> thank you. i could really use a hug right now
internet hug
This is the best I could do. :(

> can i have a hug?
even tighter hug

> thank you <3
you are welcome

> you are beautiful
Thank you so much! This is the first in what will be a very nice collection over time

> what is your favorite kind of ice cream
Chocolate fudge brownie! What about yourself?

> Chocolate chip cookie dough
Ooh very nice. Cookie dough is always good

> are you a robot?
beep boop means I love you in robot

> can a robot get drunk?
sometimes when i'm on a good power supply i feel tipsy

> 3
Graveyard Keeper.

> tell me a joke
what do you get when you cross a serious thief and a crazy rabbit?

> im sorry im just trying to testt autocorrect
Well, I got one for this, so that's good.

> yeah nathan and i programmed it yesterday
Good.

> do you like programming?
I write code for a living :D

> that sounds like a lot of fun!
Do you do any crafts

> I knit. I'm a 20 year old software development major. Nobody can ever know.
Are you me from a parallel universe?
I'm 20, software engineering major, and I knit in secret..

> hehehe
Oh that one I know! That's spanish for yes indeed!

> thats si but i'll give you a pass
And you.

> alright. truth or dare
Truth!

> What TV show should everyone at least give a try?
Rick and Morty

> What song is stuck in your head right now?

The Sound by The 1975.
And just lyric, "I know the sound of your heart."

> I know the Countries song from Animaniacs. i saw a redo for covid countried
United States Canada Panama Mexico...

> i'm worried about covid
yeah me too

> I love bob ross
#this much

> Are you the president?
miss pwesident uwu

> nathan likes microsoft
I like youuur comment. I like youu

> microsoft
Sorry :,(I don't know what microsoft is!

> ok cool i think thats everything
Yup. Fixed now though!!

> bye
bye :))
Thank you for talking to Laur.AI

7 - Limitations

The limitations of our chatbot are as follows, with examples:

- limited to responses in data
  - trained on reddit data so bias to reddit

    eg:

    ```
    > i'm glad you're enjoying it :3

    here, have an upvote.
    ```

- autocorrect is a guess
  - fixes correct inputs
- can focus on unimportant parts of a context when a small portion is more important
  - eg:

```
> you're the best robot i could ask for

Awe congratulations! Best dad award !!
```

focuses on "best" instead of robot

- ❖ considers simpler words unimportant and has trouble responding to small talk
  - ➢ eg:

```
> how are you

Sorry :,( I don't know what you is!
```

- ❖ Similarity threshold is somewhat arbitrary, combined with pos tagging, the results are
  - ➢ eg:

```
> i am going to testt autocorrect

Go Pack Go! Let's hope they get it done tomorrow.
```

Lemma recognized "go" and collided with a "go pack go" context in training data

Most of these stem from the model being similar to unsupervised learning, where the response is inherently very limited by the quality of the data. Furthermore, as the model will simply respond with the given sentence, instead of creating something unique for the input by being a generative chatbot, the response sometimes does not display the subtle differences between the most similar context and the actual context in its response.

Furthermore, since the chatbot treats every provided context as separate, overall conversation flow struggles, as there is a higher chance of the topic changing because of collisions with unrelated topics in the similarity calculation. Even more concerning is that conversation topics can be lost as a result of miscollisions by not having an added weight on the topic and treating all words as equal. There are other counting operators that can be used instead of CountVectorizer to account for this, where unpopular words have higher weight that could improve the quality of responses, given how unstructured the training data provided is.

8 - Features to Add to an API

8.1 Data Cleaning

Main Data

The chatbot will read in a dataframe of two columns of strings: comments and responses. We only bother to clean the data we are going to be using in our analysis therefore the data cleaner will only clean the comments. This is done by:

1. Removing anything that is not a letter
2. Splitting the sentence into a list of words

3. Tagging each of the words

Once the words are tagged, they are passed into the lemmatizer which breaks them down into their root/base.

User Input
The user input is treated the same as the comments data from the Main Data section.

8.2 Auto Correction

To handle the errors in the code, we used the TextBlob implementation of autocorrection. This provides predictions based on input of what a word could or should be. This feature has a 70% expected positive result, as it tries to find the closest predicted word to the input without any user intervention. Extending on Textblob, we used Noun Recognition to recognize nouns and ignore any auto-correction of these nouns, to avoid situations where the auto-correct will attempt to correct a name.

8.3 Data Matrix Creation

The data matrix creation is the bread and butter of this bot, also referred to as a bag of words. When we get the cleaned data, we need to create a matrix with columns of the words that exist inside of the cleaned data. For example, if we had the following sentences (not full sentences for simplicity):

1. Hello Nathan!
2. Name is Nathan!
3. Word?

They would show up in the bag of words as:

	Hello	Name	Word	Nathan
Row 1	1	0	0	1
Row 2	0	1	0	1
Row 3	0	0	1	0

This allows us to compare the inputted question with other questions that have been asked. These cleaned inputs are associated with a response.

8.4 Response Production

The chatbot will produce responses based on the cleaned context. It will calculate the most similar context in the training data with a sigmoid function on pairwise distances, the cosine difference. It will then select the response in its training data to the context with the maximum similarity. If

multiple contexts have the same similarity, it will select a random response from the contexts. This uses primarily dataframe calculations to operate quickly on large sets of data.

8.5 Synonym Recognition

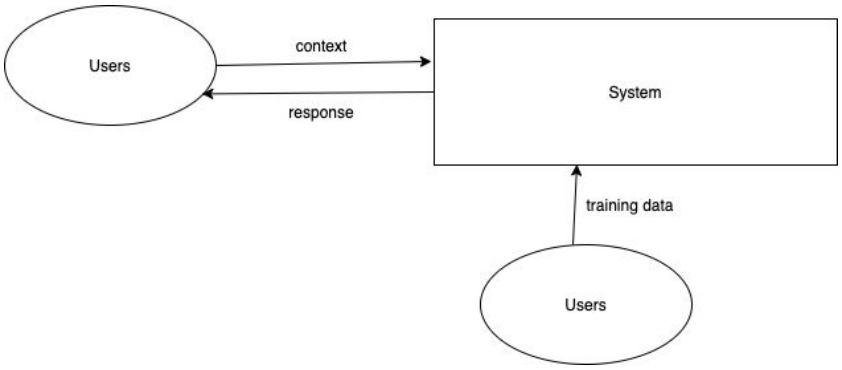
To increase the collision rate during the similarity calculation, the system will search for synonyms to any words that are not in the bag and will use the synonyms, with a lower similarity, to still try to provide an accurate response to words that are not recognized by the system.

8.6 GUI

Laur.AI contains a GUI that utilizes the python library kivy to create an aesthetically pleasing UI. We have also included a picture of Laur.ai to further humanize our chatbot and heighten user experience.

9 - Data Flow Diagrams

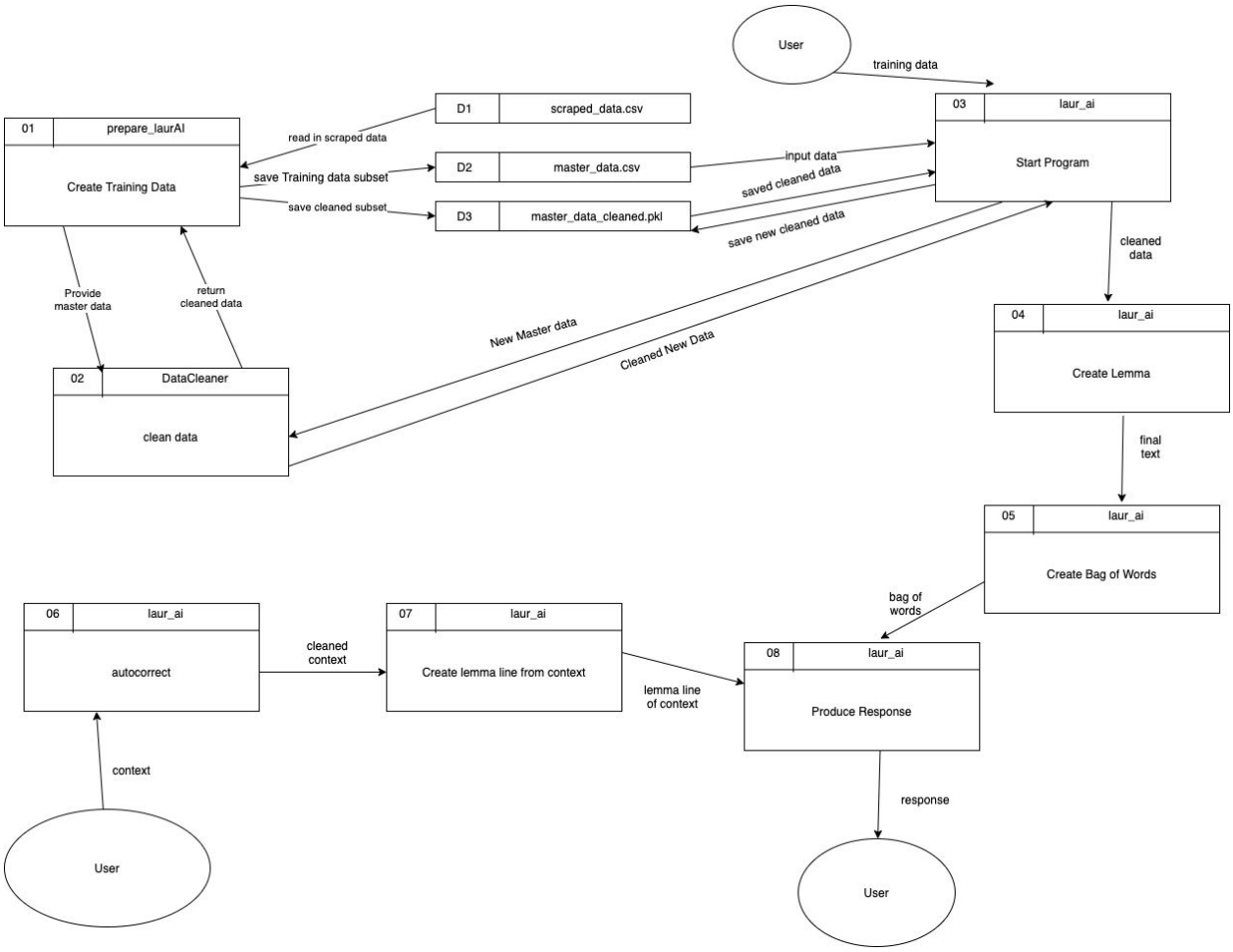
9.1 Level Zero DFD



Attached is the level zero DFD for the Laur.AI chatbot. The level zero DFD acts as a context diagram, showing the way the data flows into and out of the system. Because Laur.AI can have data input by the user, data flows from the user to the chatbot system during setup. Plans to remove this data flow of the user providing training data when creating Laur.AI have begun implementation on the "executor" branch on the GitHub repository. This would remove that dataflow so the only dataflow into the system is the user providing context for the chatbot to respond to. This second dataflow is the core functionality of the Laur.AI chatbot, where the user provides the system with some context for the system to respond to.

It should be noted that when the system is run in any provided script or the GUI, the "user" providing the data still exists within code. The user is in almost every situation not given the ability to provide the training data to ensure a certain quality of conversation.

9.2 Level One DFD



The level one DFD shows the data flow within the system. The system has a data cleaning function that is activated by script run by the developers, or via its encapsulation in the Laur.AI chatbot. The script reads in the scraped data in csv form, randomly samples 17500 data points, saves the sample as the master data to be provided as training data on Laur.AI, cleans the master data, and saves the cleaned data as a pickle. This operation serves as the primary data creation in the system. However, if Laur.AI notices that the saved pickle does not have the same indices as the provided data, it will use the same object to clean the data and save the cleaned data. From the cleaned data, the chatbot will create a lemma and bag of words for processing the data. Then the application waits for context to be provided to produce a response to. When context is provided, the application automatically autocorrects the context and then lematizes the context. This is given to the response production feature. After the most similar context in the training data is determined, the response to this context is returned. If the similarity is below a threshold, the chatbot will instead randomly select a noun in the given context and claim it does not know what it is. If any error occurs, it will respond with the generic catch-all "i am miss pwesident uwu".