**Siebert (Deep Learning) Chatbot:**

1. **Setup Environment:**
   * Install Python 3.7.9 or higher.
   * Create a virtual environment (optional but recommended).
   * Install Rasa: pip install rasa.
2. **Initialize Rasa Project:**
   * Run: rasa init --no-prompt to create a new Rasa project.
3. **Install Transformers:**
   * Since Siebert relies on deep learning models, you'll need to install the Transformers library: pip install transformers.
4. **Implement Siebert Sentiment Analysis:**
   * Update the actions.py file to include the Siebert sentiment analysis logic using the Transformers library.
5. **Train and Run the Chatbot:**
   * Train the chatbot: rasa train.
   * Run a separate terminal for action functions, (rasa run actions)
   * Start the chatbot server: rasa run.
6. **Interact with the Chatbot:**
   * Once the server is running, interact with the chatbot through the Rasa shell or integrate it with other platforms.

**TextBlob Chatbot:**

1. **Setup Environment:**
   * Ensure Python 3.7.9 or higher is installed.
   * Create a virtual environment (optional).
   * Install Rasa: pip install rasa.
2. **Initialize Rasa Project:**
   * Run: rasa init --no-prompt to create a new Rasa project.
3. **Install TextBlob:**
   * Install the TextBlob library: pip install textblob.
4. **Implement TextBlob Sentiment Analysis:**
   * Update the actions.py file to include the TextBlob sentiment analysis logic.
5. **Train and Run the Chatbot:**
   * Train the chatbot: rasa train.
   * Start the chatbot server: rasa run.
6. **Interact with the Chatbot:**
   * Interact with the chatbot through the Rasa shell or integrate it with other platforms.

**Vader Chatbot:**

1. **Setup Environment:**
   * Ensure Python 3.7.9 or higher is installed.
   * Create a virtual environment (optional).
   * Install Rasa: pip install rasa.
2. **Initialize Rasa Project:**
   * Run: rasa init --no-prompt to create a new Rasa project.
3. **Install NLTK and Vader:**
   * Install NLTK: pip install nltk.
   * Download NLTK data: python -m nltk.downloader vader\_lexicon.
4. **Implement Vader Sentiment Analysis:**
   * Update the actions.py file to include the Vader sentiment analysis logic using NLTK.
5. **Train and Run the Chatbot:**
   * Train the chatbot: rasa train.
   * Run a separate terminal for action functions, (rasa run actions)
   * Start the chatbot server: rasa run.
6. **Interact with the Chatbot:**
   * Interact with the chatbot through the Rasa shell or integrate it with other platforms.

By following these steps, we should be able to set up and run each chatbot with the specified sentiment analysis model. Make sure to update the actions.py file with the appropriate logic for each model.

**Greeting**

**Chatbot:**

"Hello! It's great to see you. How's your day going?"

**User:**

"Hi! I'm doing well, thank you."

**Explanation:**

* The chatbot initiates the conversation with a friendly greeting, setting a positive tone for the interaction.
* This greeting message is implemented using the **utter\_greet** action defined in the **actions.py** file.
* Upon receiving the greeting, the user responds with a polite acknowledgment of their current state, indicating that they are doing well.

**Emotional Check**

**Chatbot:**

"That's wonderful to hear! How are you feeling today?"

**User:**

"I'm feeling great!"

**Explanation:**

* After the initial greeting, the chatbot acknowledges the user's response and proceeds to inquire about their current mood.
* This interaction demonstrates the chatbot's ability to engage with the user on an emotional level, showing empathy and interest in their well-being.
* Depending on the sentiment detected from the user's response (in this case, "I'm feeling great!"), the chatbot selects an appropriate response using either the **utter\_feeling\_good** or **utter\_feeling\_sad** action.
* The chatbot listens for the **feeling\_good** or **feeling\_sad** intents triggered by the user's emotional state, allowing it to tailor its response accordingly.

**Game Proposal**

**Chatbot:**

"Awesome! Since you're feeling great, how about we play a fun game together?"

**User:**

"Sure! What game do you have in mind?"

**Explanation:**

* After the emotional check, if the user is feeling positive, the chatbot proposes playing a game together.
* The chatbot initiates the game proposal using the **utter\_propose\_game** action, inviting the user to participate in a fun activity.
* Upon receiving the proposal, the user expresses interest in playing the game, indicating their willingness to engage further.
* The chatbot listens for the **affirm** or **deny\_game** intents triggered by the user's response, allowing it to determine whether the user wants to proceed with playing the game.

**Game Interaction (Image-Based)**

**Chatbot:**

"Awesome! Let's play a game. I'll show you an image, and I'd like you to describe what you see or how it makes you feel. Ready?"

**User:**

"Ready!"

**Chatbot:**

"Great! Here's the first picture. What do you think about this picture?"

[The chatbot displays an image of a colorful sunset over a calm ocean]

**User:**

"It's breathtaking! The colors are so vibrant, and the reflection on the water is mesmerizing. It makes me feel peaceful and inspired."

**Chatbot:**

"Beautifully described! Now, here's another picture. What idea did you get from this picture?"

[The chatbot displays an image of a cozy cabin in a snowy forest]

**User:**

"It looks like the perfect setting for a cozy winter getaway. I can almost imagine sitting by the fireplace with a cup of hot cocoa, surrounded by the peacefulness of the snow-covered trees."

**Explanation:**

* In this example, the chatbot engages the user in a game where they interact with images instead of words.
* The chatbot prompts the user to describe their thoughts, feelings, or ideas inspired by each image displayed.
* The user responds to each image prompt by providing their interpretation or reaction, fostering creativity and engagement.
* The chatbot listens for the user's responses and may store them for further analysis or display to the parent/guardian.
* This game interaction offers a visually immersive experience, encouraging users to explore their imagination and express their thoughts in response to different visual stimuli.

**Game Interaction with Sentiment Analysis (Domain-Specific)**

**Chatbot:**

"Yay! Let's do it! Here's the first picture. What do you think about this picture?"

[The chatbot displays the first picture]

**User:**

"It's stunning! The colors are so vibrant, and the scenery is breathtaking."

**Chatbot:**

"Keep going! What else do you see?"

**User:**

"I notice the serene calmness of the water and the reflection of the sky. It makes me feel at peace."

**Chatbot:**

"Buddy, how does this picture make you feel in your thoughts?"

**User:**

"It evokes a sense of tranquility and relaxation. I wish I could be there right now."

**Chatbot:**

"Here is another picture, what do you think about this picture?"

[The chatbot displays the second picture]

**User:**

"This picture is magical! The snow-covered trees and the cozy cabin create a warm and inviting atmosphere."

**Chatbot:**

"Please, keep going! What else do you see?"

**User:**

"I imagine myself sitting by the fireplace, wrapped in a blanket, and enjoying the beauty of the winter landscape."

**Chatbot:**

"Buddy, what idea did you get from this picture."

**User:**

"I envision a peaceful retreat in nature, away from the hustle and bustle of everyday life."

**Chatbot:**

"Thanks a bunch for playing and sharing, pal! Your ideas were awesome. You're a great teammate!"

**Explanation:**

* In this conversation flow, the chatbot uses the provided replies to prompt the user for responses during the game interaction.
* The chatbot encourages the user to describe their thoughts, feelings, and ideas about the displayed images using the provided prompts.
* By incorporating these specific replies, the chatbot maintains a consistent and engaging dialogue with the user, fostering a positive interaction experience.

Top of Form

**Interaction with Parents**

**Chatbot:**

"I've stored your child's responses. Shall I also show their emotions?"

* "Your child's responses are saved. Do you want me to display their emotions?"
* "The responses from your child are logged. Would you like me to include their emotions?"
* "Your child's answers have been recorded. Should I add their emotions too?"
* "Your child's input has been saved. Would you like to see their emotions as well?"
* "I've noted your child's responses. Do you wish to view their emotions alongside?"
* "The responses from your child have been logged. Shall I capture their emotions too?"
* "Your child's answers are saved. Do you want me to capture and display their emotions?"
* "Your child's input has been noted. Shall I also record their emotions for you?"
* "Your child's responses are stored. Should I include their emotions in the display?"

**Parent:**

"Yes, please show their emotions."

**Chatbot:**

"Sure! Here are the emotions associated with your child's responses:"

* (Displays sentiment analysis results for each response)

**Explanation:**

* The chatbot prompts the parent to indicate whether they want to see their child's emotions associated with the responses.
* If the parent agrees, the chatbot displays the sentiment analysis results for each of the child's responses.
* This interaction allows parents to gain insights into their child's emotional state and engagement with the chatbot's activities, fostering transparency and communication between parents and children.