ChillPills:

A Mental Health Support Chatbot

Objective:

The project seeks to develop **ChillPills**, a mental health support chatbot that leverages **Google Dialogflow CX** to respond to emotional states such as anxiety, stress, and sadness. By integrating meditative techniques, the chatbot aims to offer compassionate and conversational emotional support, guiding users through mindfulness practices to promote well-being.

Step 1: Understanding the Theory of Emotions

To ensure that ChillPills can effectively address and respond to a variety of emotional states, we started by exploring foundational theories of emotions. One of the most influential frameworks for understanding human emotions is the *Theory of Emotions* proposed *by Paul Ekman*.

Paul Ekman, a pioneering psychologist, developed a model based on the idea that certain emotions are universal across human cultures. According to Ekman, these primary emotions are biologically programmed and have adaptive functions that aid in human survival and social interaction. Understanding these emotions and their expressions will allow ChillPills to identify and respond appropriately to the emotional states of users.

Primary Emotions

Ekman identified a set of primary emotions that are universally recognized across cultures. These include:

1. Anger
2. Fear
3. Sadness
4. Joy
5. Surprise
6. Disgust

Secondary Emotions

In addition to these primary emotions, Ekman noted the presence of secondary emotions, which are more complex and often result from the interplay of primary emotions, personal experiences, and social context. Examples include:

- *Anxiety*: Often a combination of fear and sadness, rooted in the anticipation of negative outcomes.
- *Jealousy*: A mixture of fear and anger, related to insecurity about relationships.

- *Shame*: A secondary emotion stemming from an internal conflict between personal standards and perceived social acceptance.
- *Hope*: A positive emotional state linked to aspirations, often involving the anticipation of joy or success.
- *Remorse*: The combination of sadness and guilt, usually following a mistake or moral misstep.

These secondary emotions reflect a deeper emotional complexity and may require nuanced responses that ChillPills can be trained to identify and address.

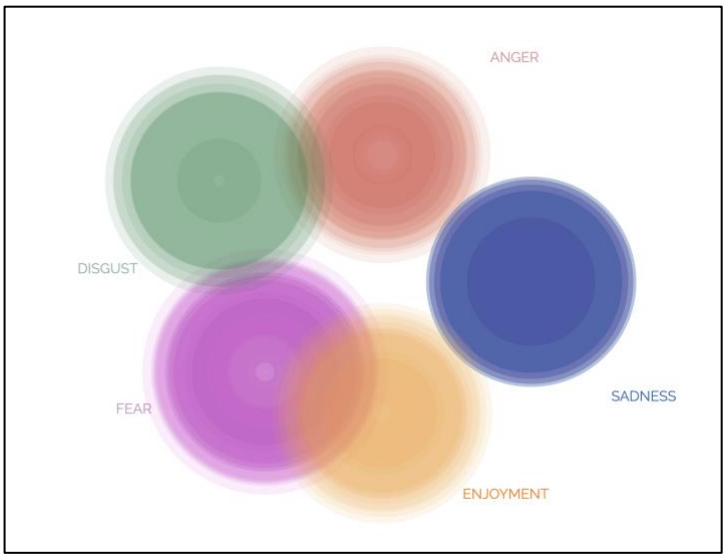
PRIMARY EMOTION	*SECONDARY EMOTION*
Joy	Hopeful, proud, excited, delighted
Fear	Anxious, insicure, inferior, panic
Anger	Resentment, hate, envy, healous, annoyes
Sadness	Shame, neglectful, depression, guilty, isolated
Surprise	Shocked, dismayed, cofused, perplexed
Disgust	Repulsed, revolted, nauseated, intolerance

Step 2: Incorporating Ekman's Framework into ChillPills Chatbot Design

Given the insights from Ekman's Theory, the chatbot will be designed to identify and categorize user emotions based on their textual input. By leveraging natural language processing (NLP) techniques and sentiment analysis, ChillPills will analyze the sentiment behind each user message to detect primary and secondary emotions.

For the

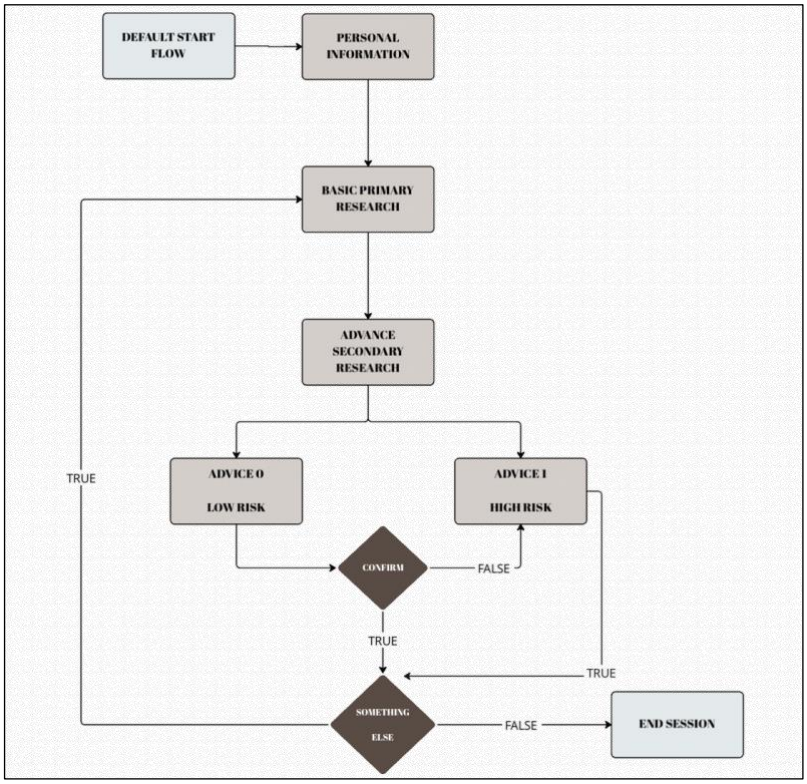
- *Emotion Detection*: The chatbot will use NLP algorithms to identify keywords and patterns in the user's input. For example, phrases like "I feel frustrated" or "I can't take this anymore" would trigger the detection of anger, while phrases like "I feel overwhelmed" or "I just want to give up" might indicate sadness or anxiety.
- *Response Generation*: Based on the detected emotion, the chatbot will provide personalized responses. If a user expresses anger, the chatbot may offer calming techniques such as deep breathing or mindfulness practices. For users expressing sadness or anxiety, the chatbot could suggest meditative techniques or offer empathetic, reassuring responses.



https://atlasofemotions.org/#continents/disgust

For the realization of this project was used_Dialogflow CX is an advanced conversational AI platform that enables the creation of dynamic and interactive chatbots. It uses a state-based system to manage complex dialogues, ensuring a structured flow of interactions. With support for multiple channels and integration with external systems, it provides scalability and adaptability. Its machine learning capabilities enhance natural language understanding, allowing for more accurate and context-aware responses.

Flow chart



1. *Default Start Flow*:
2. *Personal Information*:

   The first phase of the process involves gathering personal information. This section is crucial for collecting initial data.
3. *Basic Primary Research*:

   After obtaining personal information, the primary emotions of the user are collected and examined. These emotions are essential because they provide a direct understanding of the user's initial emotional state and perceived risk.
4. *Advanced Secondary Research*:

   Next, secondary emotions are explored, which are more complex and reflective emotions that build upon the primary emotions.
5. *Risk Decision (Advice 0 / Advice 1*):

   After examining the primary and secondary emotions, a risk assessment is made, which determines the advice to provide to the user:
   - Advice 0 (Low Risk): If the risk assessment indicates low perceived risk, appropriate advice is given for low-risk situations.
   - Advice 1 (High Risk): If the risk assessment indicates high perceived risk, advice is provided for dealing with high-risk situations.
6. *Confirmation*:

   After giving the advice, the user's response is checked:
   - Yes: If the user confirms, the process proceeds to the next phase.
   - No: If the user does not confirm, the flow returns to a previous phase for further evaluations or modifications.
7. *Something Else*:

   In this step, the user approaches the end of the session, either accepting the final advice or starting a new session, if necessary.
8. *End Session*.

Step 3: Synonym' Generation

The next step is to create a dataset that helps the chatbot better understand the language variations related to each emotion.

Each emotion has been associated with a sequence of synonyms divided into categories (nouns, verbs, adjectives, adverbs). For example, for anger emotion we could include synonyms like furious (adjective), irritated (verb), rage (noun), and angrily (adverb).

In our implementation of DialogFlow CX, we have added Primary Emotion and Secondary Emotion as custom Entity Types to better capture and understand the emotional state of users during interactions.

Entities represent categories of objects, concepts, or values within a conversation that the system can identify to better understand the user's intent and personalize the response.

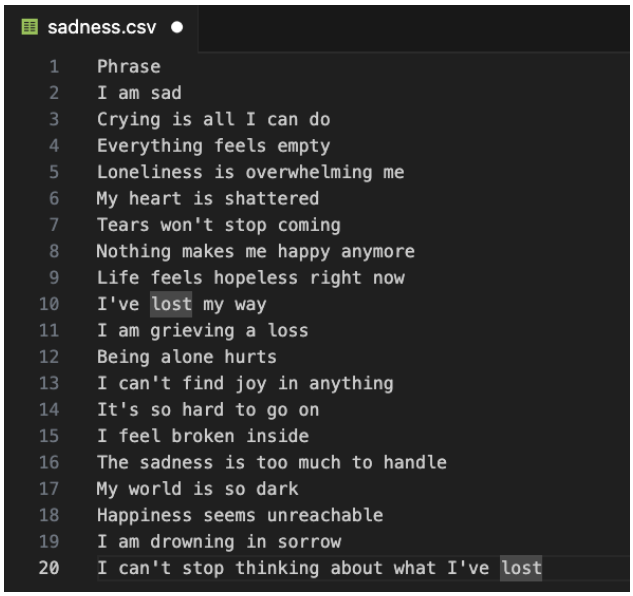
After generating the synonyms for each emotion, we organized them into JSON files, which were then uploaded to the DialogFlow CX platform.

Step 4: Dataset Creation

The next step was to create at least 30 sentences for each emotion.

The sentences were generated by ChatGPT, which used different synonyms and formulations for each emotion

These phrases have been organized based on primary and secondary emotions, creating a variety of expressions to train the chatbot. Each emotion has a set of phrases that the chatbot will use to recognize users' emotional states and respond appropriately.



After creating the dataset in JSON format, we integrated it into the **intents**. Each intent is an object that associates a group of example phrases, known as "training phrases", with a specific response from the system. When the user provides input, DialogFlow CX analyzes the phrase and attempts to match it to the most relevant intent, determining the most appropriate response based on natural language analysis. This process allows the system to understand the user's intent and respond in a accurate manner.

Step 5: Future Integration of Meditative Techniques

To support users through their emotional challenges, ChillPills will integrate a range of meditative techniques that can help reduce stress, anxiety, and negative emotions:

- *Breathing Exercises*: Guided breathing practices to help calm the nervous system and reduce anxiety.
- *Mindfulness Meditation*: Techniques that encourage present-moment awareness to combat stress and rumination.
- *Progressive Muscle Relaxation*: A method to release physical tension and promote relaxation.
- *Visualization Techniques*: Guided imagery to help users feel more grounded and relaxed.

Each technique will be introduced based on the user's emotional state, helping to restore balance and emotional well-being.

CURIOUS LIMITATION:

1. *Limit in the Analysis of Sentiment*

In the context of the implementation of DialogFlow CX, we found a problem related to the analysis of sentiment, due to the polysemy of the term, for example "upset".

In many situations "upset" is associated with emotions such as disgust or fear, so it has nuances that can represent more than one emotion.

- Disgust: In some cases, "upset" can be used to express a reaction of disgust or repulsion, such as when someone reacts negatively to something unpleasant or repugnant.

Example: "The idea of eating that spoiled food really upset me." (In this case, "upset" is related to a negative reaction similar to disgust.)

- Fear: In other cases, "upset" may describe an emotional reaction of concern, anxiety or fear in the face of a situation perceived as threatening or disturbing.

Example: "She was upset after hearing the news of the accident." (In this case, "upset" may reflect an emotional reaction of fear, anxiety or concern.)

DialogFlow CX struggles when a term has multiple emotional connotations. "Upset" is an example of a term that can express two contrasting emotions (disgust and fear), but the system is designed to detect only one type of emotion per word, thus unable to distinguish all the emotional nuances. This limitation means that in some contexts, DialogFlow CX could incorrectly classify a conversation or user response. The platform might mistakenly associate a response with just one emotion (e.g., only disgust or only fear), whereas the user may actually be experiencing a combination of emotions or a more complex emotional response.

2. *Complexity of secondary emotions*

Secondary emotions, which are mixtures of primary emotions (e.g. anxiety from fear and sadness, or jealousy from fear and anger), can be difficult to identify for ChatGPT. Although the model is able to recognize primary emotions such as anger or sadness, interaction between different emotions can confuse the model and lead to inaccurate responses.

For example: Anxiety, which is a combination of fear and sadness, may be difficult to detect if the sentences do not contain direct expressions of these emotions.

In addiction, ChatGPT has difficulty recognizing complex or subtle emotions, such as ambivalence (feeling happy and sad at the same time) or mixed emotions.

A sentence like "I'm glad I finished, but I also feel sad that it's finished" might be difficult to analyze properly, since the model could detect only one predominant emotion, without grasping the emotional conflict.

3. *Insufficient documentation*

Another significant limitation in the implementation of DialogFlow CX is the unclear documentation and the lack of available support material. Although Google Cloud provides official documentation, it can be incomplete, insufficiently detailed, or difficult to follow in some critical aspects, especially for beginners or complex projects.

4. *Failure to Synchronize Collaborative Work*

An additional limitation encountered when using DialogFlow CX concerns real-time collaboration between multiple developers. When two users work simultaneously on the same project, changes saved by one user may not be immediately visible to the other.

This issue appears to be related to the browser's cache and cookie management. DialogFlow CX uses cookies and locally stored data to optimize the interface loading process. However, in a collaborative context, this mechanism can cause synchronization issues, as changes made by one user are not updated in real time on the other user's interface.

As a result, developers need to manually clear browser cookies and cache every time to see updates, slowing down the workflow and reducing collaboration efficiency.

5. *Manual Deletion of Terms in JSON File*

Another issue encountered during the implementation of DialogFlow CX concerns the management of terms associated with emotions when deleting a word from the platform's user interface. When a term is manually removed from the interface to simplify emotion mapping (for example, to keep a word associated with only one emotion), one would expect this change to be automatically reflected in the JSON file generated by the platform.

However, DialogFlow CX does not automatically delete the term from the JSON file when it is removed from the interface. The term deleted from the interface remains defined in the JSON file, causing the system to continue recognizing it, which may lead to incorrect classifications during sentiment analysis. This behavior creates a discrepancy between the user interface and the exported JSON file content, potentially resulting in misconfigurations and unexpected behavior in the system.

Conclusion

After an in-depth analysis of the implementation process for the ChillPills mental health support chatbot, we identified several critical limitations that hinder its development and long-term viability. Key issues include ambiguities in sentiment analysis, difficulty managing complex emotional nuances, insufficient documentation, real-time collaboration challenges, and manual synchronization issues related to the management of emotional terms in JSON files.

These limitations significantly impact the chatbot's ability to provide accurate emotional recognition, seamless collaboration, and efficient project maintenance. Given the sensitive nature of mental health support, where precision and responsiveness are essential, these constraints pose substantial risks to the reliability and effectiveness of the final product.

Considering these factors, we have decided to halt the development of the ChillPills project. We believe that moving forward would require additional platform enhancements or the adoption of a more suitable technology capable of managing complex emotional models and real-time collaboration efficiently. This decision reflects our commitment to ensuring that any mental health support tool we develop meets the highest standards of accuracy, reliability, and user safety.

WordNet:

A Study of Sentiment Analysis

Objective:

Emotions play a crucial role in human communication and language understanding.
The project described in this report aimed to create a WordNet specialized in emotions in the Italian languag[1]e, designed to support sentiment analysis and word sense disambiguation. The initiative arose from the need to structure and accurately represent the complexity of emotional states, providing a useful tool for natural language processing (NLP).
In addition to building the WordNet, an interactive application was developed to enable dynamic exploration, making it easier to understand semantic relationships between emotional terms.

The Concept of WordNet and Its Role in Semantic Disambiguation

A WordNet is a semantic network that organizes lemmas into synonym groups, establishing relationships between them. This type of resource is widely used in computational linguistics, as it enhances the ability of NLP systems to determine the precise meaning of a word within a given context. Semantic disambiguation is particularly useful in sentiment analysis because it helps distinguish different emotional nuances that can affect the interpretation of a text.
Unlike existing WordNets, this project focuses exclusively on emotions and affective states, with particular attention to the Italian language. The decision to develop a specific WordNet for Italian stems from the lack of similar resources in this language, which is often studied less thoroughly compared to English. As a result, sentiment analysis in Italian tends to be less accurate than in other languages, making it necessary to create a tool that bridges this gap.

Step 1. Structure of the Emotions WordNet

To ensure easy integration with natural language processing applications, the WordNet was created in JSON format. Each emotion is represented as an object within a hierarchical structure that includes synonyms, antonyms, hyponyms, hypernyms, a textual description,
and a set of related emotions.

[1] For more details and access to the project, visit the **GitHub repository***:*
https://github.com/Chiaramartina/ChillPills/tree/main/progetto_wordnet

Formalized JSON Structure

This structure ensures a clear and expandable semantic mapping, facilitating the management of emotions and their relationships.

Step 2. Process of Identifying and Mapping Emotions

The selection of emotions included in the WordNet was guided by psychological studies and lexicographical sources to represent a wide range of emotional states. Both *primary emotions*, which form the foundation of human emotional experience (e.g., joy, sadness, anger, fear), and *complex emotions*, which arise from combinations of primary states or social interactions (e.g., gratitude, envy, melancholy), were incorporated. This distinction allows the mapping of both fundamental emotional responses and more nuanced human feelings.

Each term was analyzed to establish significant semantic relationships. The main categories used include:

- **Synonyms**: Words with similar or equivalent meanings. For example, *happiness* is a synonym of *joy*, while *melancholy* can be considered a synonym of *sadness*.
- **Antonyms**: Opposite emotions on the same conceptual axis. For example, *joy* is the opposite of *sadness*, *anger* is the opposite of *calm*, and *love* is the opposite of *hate*.
- **Hyponyms**: More specific emotions belonging to a broader category. For example, *enthusiasm* and *satisfaction* are hyponyms of *joy*, while *nostalgia* is a hyponym of *sadness*.
- **Hypernyms**: More general emotions that encompass other emotions. For example, *positive emotion* is a hypernym of *joy*, while *negative emotion* is a hypernym of *sadness* and *fear*.

- **Related Emotions**: Emotional states that are not synonyms or antonyms but have a conceptual connection or frequently occur together. For example, *hope* is related to *joy*, while *shame* is often associated with *sadness*.

Each emotion was then accompanied by a textual description clarifying its meaning, providing a structured and easily navigable semantic framework.

Step 3: Development of the Interactive Visualizer

After constructing the emotions WordNet, an interactive application was developed to intuitively explore the semantic network of emotions. The visualizer allows users to load a JSON file containing the WordNet, select emotions to analyze, generate an interactive network of term connections, and view details about displayed emotions.

The application follows the *Model-View-Controller (MVC***)** architecture and consists of the following main files:

- *main.py* – Manages application startup.
- *controller_model.py* – Implements the application's controller, coordinating data management and user interaction.
- *emotion_view.py* – Manages the main window, emotion selection, and the interactive network visualization.
- *splash_view.py* – Manages the splash screen with options to start the application or load a new WordNet.

Step 4. Application Features

Loading the WordNet

At startup, an initial splash screen appears, offering the option to load a JSON file containing the WordNet. The user can select a custom file via a file picker or use the default WordNet already present in the application's directory. The controller verifies the correctness of the file and loads it into the model for use in network visualization.

Emotion Selection and Network Generation

After loading the WordNet, the user can select one or more emotions from a list on the left side of the interface. By pressing the "Generate Network" button, the system creates an interactive network of the selected emotions, highlighting semantic connections between terms.

The network is visually represented using colored nodes and edges:

- *Nodes* represent emotions.

- *Edges* indicate semantic relationships (synonyms, antonyms, hyponyms, hypernyms, related terms). Each relationship has a specific color for easy identification.

The generated network is saved as an interactive HTML file and displayed within the application using an integrated web viewer.

Emotion Highlighting and Search

A search bar at the top of the window allows users to quickly find an emotion within the network. When a word is entered and the search button is pressed, the corresponding node in the network is highlighted with a different color, making it immediately visible. If the searched emotion is not present in the generated network, a warning message is displayed.

Relationship Legend

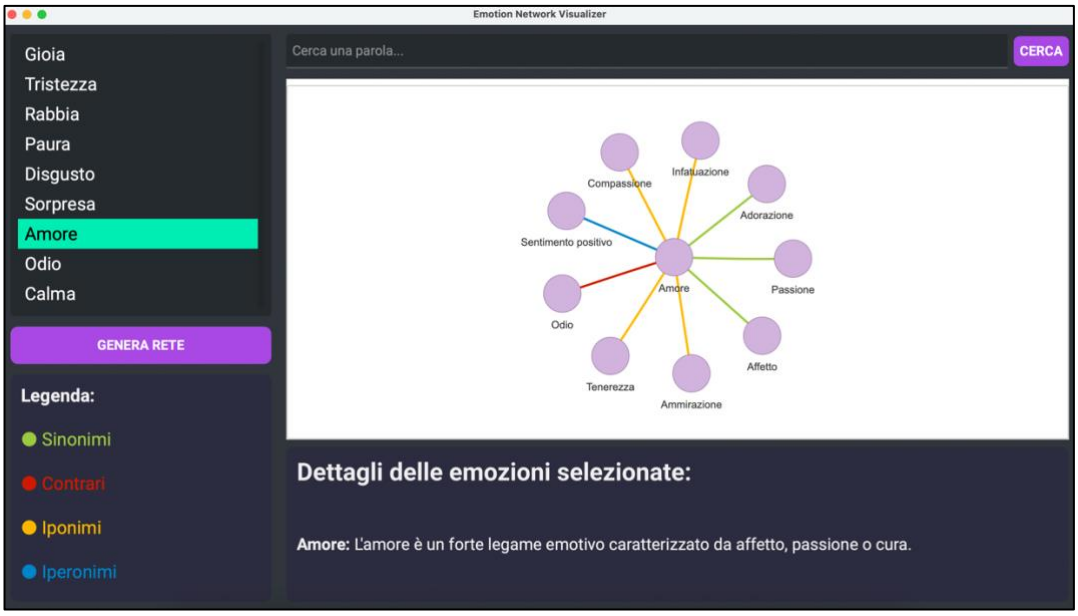
On the right side of the interface, a legend provides information about the colors used in the network to represent different semantic relationships between emotions, helping users interpret the structure of the generated network easily.

Emotion Details

Below the interactive network, a dedicated details section displays textual information about selected emotions. For each emotion, the following details are shown:

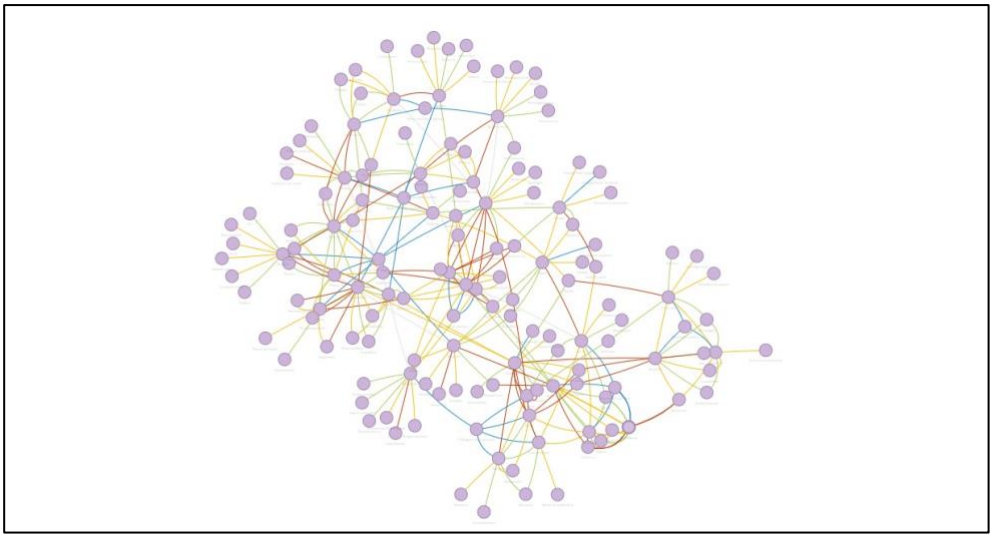
- A brief description of its meaning.
- Related emotions.
- Synonyms and antonyms.

This section helps users better understand the semantics of each term and its contextual relevance.



The app displays a selected emotion along with its related network visualization.

It is also possible to select all emotions at once and generate a complete WordNet, visualizing the full network of emotional relationships.



The complete network

Step 5. File Structure

Model (controller_model.py)

The model is responsible for loading and managing the emotion-related data. It reads the JSON file containing the WordNet and organizes the information into a data structure accessible by both the controller and the view.

View

- *splash_view.py*: Manages the initial splash screen with buttons to start the application, load a new WordNet, and display information.
- *emotion_view.py*: Contains the main application window, including the list of emotions, the interactive network, the search bar, the legend, and the details section.

Controller (controller_model.py)

The controller is the core of the application and handles the logic and interactions between the model (which contains the WordNet data) and the view (the graphical user interface). It is implemented in the controller_model.py file through the MainController class, which orchestrates data loading, interface management, and interactive semantic network generation.

WordNet Loading Management

When the application starts, the controller automatically loads a JSON file containing the default WordNet. However, users can choose to load a custom WordNet file using a file picker. The controller verifies the correctness of the JSON file and passes it to the model, which organizes it

into a data structure suitable for visualization. If the file is invalid or has formatting errors, an error message is displayed.

Interactive Network Generation

After the user selects one or more emotions, the controller constructs a semantic network representing the connections between emotional terms. The network is generated using the PyVis library, which allows for the creation of a dynamic graphical visualization. The controller assigns different colors to the network edges based on the type of relationship between emotions (synonyms, antonyms, hyponyms, hypernyms, correlations) and generates an interactive HTML file representing the semantic structure.

Visualization and Portability Management

The application uses a local HTTP server to serve the interactive network's HTML file within the view. This approach ensures greater portability, as the network visualization does not depend on external resources or internet connections but is entirely managed locally.

The controller launches an internal server on the first available port (starting from port **8000**), allowing the graphical interface to load the HTML file directly through an integrated web browser. The server runs in a separate thread, ensuring that the application remains responsive even during network generation.

This solution makes the application compatible with multiple operating systems (Windows, macOS, Linux) without requiring additional installations. The user can explore the interactive network without depending on a specific development environment or complex configurations.

Search and Highlighting Management

The controller implements a search function that allows users to quickly find an emotion within the generated network. When the user enters a term in the search bar and presses the corresponding button, the controller executes a JavaScript script directly on the displayed network, dynamically changing the node colors to highlight the searched term. If the term is not found in the generated network, a warning message is displayed to the user.

Application Closure Management

When the user closes the application, the controller ensures the proper termination of the internal server, preventing background processes from remaining open. This guarantees a clean shutdown and avoids potential conflicts with other applications that might use the same network port in the future.

Step 6. Conclusion and Future Developments

The creation of a specialized WordNet for sentiment analysis in Italian and its interactive visualizer represents a significant advancement for computational linguistics in the Italian language. The WordNet provides a detailed semantic network enriched with precise relationships between emotions, while the visualizer enables an intuitive understanding of conceptual connections. Future developments could involve expanding the WordNet with new emotions and integrating it with machine learning algorithms to further refine semantic relationships. Additionally, publishing the WordNet as an API could make it accessible to researchers and developers, increasing its usefulness in sentiment analysis and NLP applications for Italian.