ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 01 January 2024

A Software Implementation and Documentation of an Affect-Aware Gateway API using PyAIML and Python Flask: Training AI Systems with Emotionally Intelligent Responses to English Idiom Queries

[1] Jonathan Bishop, [2] Wahid Hassan, [3] Robert Bilsland, [4] Jason Barratt, [5] Elias Alexander

[1] Crocels Research CIC, UK
[2] University of South Wales, UK
[3] [4] Independent Researcher
[5] Clarisa Technologies, India
Email: [1] jonathan.bishop@crocels.ac.uk, [2] devwahid5@gmail.com, [3] rcbilsland@gmail.com, [4] jwbarratt29@gmail.com, [5] rcbilsland@gmail.com

***Abstract*— This research enhances Artificial Intelligence Markup Language (AIML) Systems understanding of English idioms and their emotional contexts. By integrating a database of 3,500 English idioms with 16 emoticons, each representing different emoticons. The study aims to enable AI to interpret idioms beyond their literal meanings and to respond appropriately to their emotional undertones. The aim to enable AIML to recognise these phrases and understand their contextual meaning and emotional connotations. This understanding is crucial in rendering AIML-based interactions more natural, empathetic and effective.**

***Index Terms*— AIML, PyAIML, Python Flask, Affective Computing, English Idioms, Artificial Intelligence**

I. INTRODUCTION

In the ever-evolving landscape of Artificial Intelligence, the quest for creating systems that closely mimic human understanding and expression remains a pinnacle of technological advancement [4]. This project represents a significant stride in this direction. It focuses on imbuing AI with a nuanced grasp of English idioms, with the ability to discern and express associated emotional undertones, a feat that bridges the gap between mechanical processing and human-like comprehension. Recognising this, the project collected and compiled a diverse collection of over 3,500 English idioms. The aim was to enable AIML to recognise these phrases and understand their contextual meanings and emotional connotations. To achieve this, the authors embarked on a process of data collection. The authors used reliable online sources to gather idioms, ensuring a broad representation of cultural and linguistic diversity. Each idiom was then carefully encoded into an XML-based AIML dataset. This dataset serves as the bedrock of the authors' project, providing a repository from which the AI can draw to enhance its linguistic processing capabilities. Further, the authors integrated a set of 16 emoticons, representing a distinct emotional state, from the Crocels Troller-Sniper Emotion Index 16 [1]. The AIML parser can understand the idioms and respond in a way that is consistent with the intended emotional tone. The system advances beyond earlier systems, which searched idioms while a user was taking part in social situations [2] [3].

II. RESEARCH OBJECTIVE AND METHODOLOGY

The primary objective of this research is to enhance the capabilities of Artificial Intelligence Markup Language (AIML) systems in processing and understanding English idioms and their associated emotional contexts. This involves compiling a comprehensive list of 3,500 English idioms and integrating a set of 16 emoticons, each representing a distinct emotional state, into the AIML system. The goal is to enable the AI system to comprehend the literal meaning of idioms and to appropriately respond to them in a context-sensitive manner that aligns with the emotional undertones conveyed by the emoticons.

III. DATA COLLECTION: COMPILING A COMPREHENSIVE LIST OF 3,500 ENGLISH IDIOMS

The data collection phase of the research project involved an extensive and meticulous process of gathering a wide array of English idioms. This phase was pivotal in establishing a foundational database for enhancing the Artificial Intelligence Markup Language (AIML) system with a nuanced understanding of idiomatic expressions and their emotional connotations.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 01 January 2024

The authors initial step was to identify reliable and diverse sources from the internet that provided a rich repository of English idioms. This included exploring linguistic databases, online dictionaries, idiom-specific websites, and digital libraries. The criteria for source selection; authenticity, comprehensiveness, and cultural diversity of the idiomatic content. Sources were rigorously evaluated to ensure the authors' stringent standards for accuracy and relevance.

Once finalised, the process of idiom extraction commenced. Performed by a combination of automated and manual methods. Custom scripts were written in Python to crawl through the identified websites and extract idioms. These scripts were designed to parse web pages, identify idiomatic expressions, and retrieve them along with their meanings and usage examples. Manual intervention was essential in this phase to ensure the quality of the data. By reviewing the extracted idioms for accuracy, removing duplicates, and cross-verifying the meanings and usage contexts. The manual process also helped in discerning culturally significant idioms that automated scripts might have overlooked.

The next crucial step was the structuring and storage of the collected idioms. To achieve this, the authors opted for an XML (Extensible Markup Language) format, which offers flexibility and ease of integration with AIML systems. Each idiom carefully encoded into the XML file with its corresponding details (meaning, usage, and emotional undertones). The XML file was designed to be comprehensive yet intuitive, facilitating easy access and retrieval of idioms for the subsequent phases of the project. It included tags for categorising idioms based on their emotional connotations and usage contexts, which would later assist in pairing them with the appropriate emoticons.

To ensure the integrity and quality of the authors' idiom database, a thorough quality assurance process was implemented. Involving multiple rounds of validation where the XML file was checked for consistency, accuracy, and completeness. Anomalies or errors detected during this stage were rectified, ensuring that the database was of the highest standard. The data collection phase executed with a focus on precision, diversity, and comprehensiveness. The meticulous process of sourcing, extracting, structuring, and validating over 3,500 idioms laid the groundwork for enhancing the AIML system's capability to understand and express the emotional dimensions of language, marking a significant advancement in the field of AI and linguistic processing.

IV. CREATING THE LIST OF 16 EMOTICONS: A DETAILED EXPLORATION

The development of the emoticon list for this project was a nuanced process, intricately tied to understanding the spectrum of human emotions. The authors made a comprehensive list of 16 emoticons from the Crocels Troller-Sniper Emotion Index 16 [1], each representing a distinct emotional state or response. The focus was to ensure that these emoticons encompassed a wide range of feelings, from positive to negative, and from mild to intense.

V. DATASET: EMOTICONS AND THEIR EMOTIONS

Below is a table representing the 'Emoticons' and their associated 'Emojis', as part of the Crocels Troller-Sniper Emotion Index 16 [1].

Table 1. Emoticon/Emoji List

Emoticon	Emoji Representation	Emotions Associated
:-""	Trident Emblem	Shame, Guilty, Obscene, Hurt, Scornful, Rejected, Resent
:@	Angry Face	Anger, Enraged, Hate, Outrage, Rage, Violent
^o)	Unamused Face	Disgusted, Displeased, Ridicule
8o)	Persevering Face	Disdainful, Hostile, Jealousy, Menace, Nasty, Obnoxious
:∞	Boy	Detached, Moral, Pride, Reserved, Snob
|-)	Smiling Face With Sunglasses	Rigid, rude, selfish, serious, sceptical, though.
;-)	Winking Face	Erotic, Merry, Romantic, Sexy
:-(	Disappointed Face	Depressed, Despairing, Distressed, Gloom, Helpless, Horror, Misery, Regretful, Sad, Stress, Suicide, Unhappy, Upset, Pity
:-0	Man	Devoted, Hopeful, Repentant, Wise
8-)	Mage	Elated, Excitement, Lively, Optimism, Triumphant
:-o	Astonished Face	Grateful, Kind, Reverent, Subdued, Thoughtful, Timid, Warmth
:-)	Smiling Face	Friendly, Impressed, Natural, Nice, Pleasure, Relaxed, Satisfied
:ℵ	Ghost	Embarrassment, Startled, Insecure
:-#	Fearful Face	Afraid, Discouraged, Fear, Loneliness, Nervous, Scared, Terrified
:-D	Face with Tears of Joy	Enjoyment, Happy, Joy, Joyful, Mischief, Silly, Tease, Wit
:-|	Tired Face	Fatigued, Rusty, Sleep

Source: Adapted from Bishop [1].

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 01 January 2024

VI. EMOTICON-EMOTION ALIGNMENT

Each emoticon was aligned with its respective emotions, as defined in the Crocels TS-EI-16 [1]. This alignment was not just about selecting an emoticon but understanding the depth and variety of emotions it could represent. For example, the emoticon '8o)' (Persevering Face), linked with the character type 'Snert', was chosen for its ability to convey emotions like hostility and jealousy, essential for responses that are meant to show disdain or disapproval. The emoticon ':∞' (Boy), associated with 'Trickster' and 'Pecker', was selected for its representation of pride and moral superiority, useful in situations where the AI needs to express self-assuredness or ethical stances. The range of emotions covered by these 16 emoticons included anger, shame, disgust, pride, despair, joy, embarrassment, fear, and more, ensuring that the authors' AI system could respond appropriately to a wide spectrum of human interactions.

VII. INTEGRATION INTO THE AIML SYSTEM

The final step was the integration of these emoticons into the AIML system. Each emoticon is encoded alongside the idioms in the XML database, allowing for a dynamic and context-sensitive response mechanism. Creating the emoticon list was a detailed process that involved careful selection, emotional categorisation, and integration into the authors' AIML system. This process was instrumental in enhancing the AI's ability to communicate with a deeper understanding of human emotions, making it a significant step forward in the field of AI and emotional intelligence.

VIII. DATA IMPORT AND MAPPING USING PYTHON

Importing Idiom Data from XML

The initial step in the methodology involves importation of idiom data from an XML file. This file, meticulously compiled during the data collection phase, contains over 3,500 English idioms. Python, renowned for its data processing capabilities, is the chosen programming language for this task. To import this data, Python's `xml.etree.ElementTree` module is used. This module is adept at parsing XML files, enabling the extraction of idioms and their associated information. The process commences with loading the XML file into Python, followed by iterating over each element – in this case, idioms. Each idiom is then extracted and stored in a Python data structure, such as a list or dictionary, for easy access and manipulation.

IX. IMPORTING EMOTICONS AND EMOTIONS

Following the idiom importation, the next step is to import the emoticons and their corresponding emotions. This data, derived from the Crocels TS-EI-16 dataset, is crucial for mapping each emoticon to its respective emotional spectrum. They are then linked and imported into Python, through a CSV file or directly from a database, depending on the storage format. Python's `pandas` library, known for its robust data manipulation capabilities, is employed to handle this dataset. The library allows for efficient data import and provides functionalities for creating a structured format, such as a DataFrame. This DataFrame serves as a map, linking each of the 16 emoticons to its array of corresponding emotions.

X. CREATING EMOTICON-EMOTION MAPS

The creation of emoticon-emotion maps is a critical part of this phase. Each emoticon is mapped to its respective set of emotions, forming a key-value pair in a Python dictionary. This mapping is vital for later stages where the AI system needs to understand the emotional context of each response.

XI. GENERATING PROMPTS AND UTILISING OPENAI API

Concatenating Emoticons with Idioms

With the idioms and emoticon-emotion maps ready, Python's looping structures come into play. A loop iterates through the list of 3,500 idioms. For each idiom, another nested loop concatenates it with each of the 16 emoticons. This process results in 56,000 unique combinations – each a fusion of an idiom with an emoticon, representing a distinct emotional context.

XII. UTILISING OPENAI API'S ASYNCHRONOUS FUNCTION

The concatenated idiom-emoticon pairs are now used to prompt the OpenAI API. Given the large number of prompts (3,500 idioms x 16 emoticons), the API's asynchronous function is employed. This function allows multiple requests to be sent to the API simultaneously, significantly speeding up the process of obtaining responses. Python's `asyncio` library is used for this purpose. It enables the sending of asynchronous requests to the OpenAI API, ensuring that each idiom-emoticon pair is processed efficiently. This approach is essential to handle the sheer volume of prompts and to retrieve responses promptly.

XIII. RETRIEVING AND SAVING RESPONSES

As responses are received from the OpenAI API, they are captured and stored. Each response, aligned with a specific idiom-emoticon pair, is saved in a text file. These text files are named after the respective idioms, ensuring easy identification and retrieval. Each text file contains 16 responses, corresponding to the 16 emoticons paired with that particular idiom. This organisation method allows for a structured and accessible way to store the vast amount of data generated by the API. The process of saving these responses involves Python's file-handling capabilities. A loop iterates through the received responses, writing each one into the

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 01 January 2024

corresponding text file. Python's efficient file handling ensures that this process is executed smoothly, maintaining the integrity and organisation of the data.

This mechanism leverages Python's powerful data processing and asynchronous capabilities to handle a large dataset of idioms and emoticons. By creating unique combinations of idioms and emoticons and utilising the OpenAI API to generate responses, the authors collected a vast and diverse range of responses for this dataset.

XIV. ANALYSIS OF TEXT FILES USING PYTHON, PANDAS, AND NLP

Text File Processing and Data Cleaning

Once the text files containing the responses of idioms paired with emoticons are generated, the next crucial step is to analyse these files for data quality and relevance. Python, along with its libraries Pandas and Natural Language Processing (NLP) tools, plays a pivotal role in this phase. Initially, Python scripts are employed to load these text files. Given the extensive number of files (one for each idiom), automation is key. Python's file handling capabilities enable efficient loading of each text file into a Pandas DataFrame. This transformation is crucial for organising the data and facilitating its analysis. Once loaded into DataFrames, the data undergoes a cleaning process. This involves scanning for and removing any gibberish or out-of-place special characters. Gibberish responses, often manifested as random strings of characters or nonsensical word combinations, can skew the analysis and provide inaccurate insights. Similarly, special characters that do not contribute to the meaning or emotional tone of the response are also removed. This cleaning process ensures that the data is as accurate and relevant as possible for further analysis.

XV. SENTIMENT ANALYSIS AND EMOTIONAL ALIGNMENT

The core of this phase involves sentiment analysis, a facet of NLP. Sentiment analysis tools are used to determine the emotional tone of each response – whether it is positive, negative, or neutral. This step is critical in understanding how well the response aligns with the intended emotion of the paired emoticon. Python's NLP libraries, such as NLTK or TextBlob, are adept at performing sentiment analysis. They can evaluate the text of the responses and assign a sentiment score to each. This scoring allows us to quantitatively assess the emotional tone of the responses.

XVI. FEEDBACK LOOP FOR RESPONSE OPTIMIZATION

Comparison with Emoticon Emotions

The sentiment analysis results are then compared with the intended emotions of the emoticons. This comparison is pivotal in determining the appropriateness of the responses. If the sentiment of a response aligns well with the emotion represented by the emoticon, the response is deemed suitable. However, if there is a mismatch – for instance, a positive response where a negative one was expected, or vice versa – the response is flagged as incorrect. It helps in maintaining the integrity and reliability of the AI system in terms of emotional intelligence.

XVII. FEEDBACK LOOP FOR RESPONSE REFINEMENT

The flagged responses, identified as mismatches, feed into a feedback loop. This loop is an integral part of the response optimisation process. Using the previously established response pipeline, the AI is prompted to generate new responses for the idioms paired with the respective emoticons. The feedback loop operates iteratively. Each new set of responses generated is again subject to sentiment analysis and compared with the intended emoticon emotions. This iterative process continues until the responses align satisfactorily with the emoticon emotions, effectively weeding out inappropriate responses.

XVIII. CONTINUOUS IMPROVEMENT AND SYSTEM EVOLUTION

The feedback loop is not just a mechanism for immediate correction; it also serves as a tool for continuous improvement of the AI system. By repeatedly refining the responses, the AI learns and adapts, becoming more adept at understanding and aligning with the nuanced emotional contexts of different idiomatic expressions. The analysis of the text files using Python, Pandas, and NLP tools is a critical step in ensuring the quality and emotional accuracy of the AI responses. The sentiment analysis, combined with a rigorous feedback loop, forms a robust mechanism for refining the AI's responses.

XIX. ENHANCING AIML WITH EMOTION ATTRIBUTES AND COMPILING RESPONSES

Integrating Emotional Context in AIML

The advancement in Artificial Intelligence Markup Language (AIML) through the integration of an emotional context marks a significant enhancement in its existing format. The addition of a new attribute, 'emotion', to the AIML schema is a pivotal part of this enhancement. This attribute aligns with one of the emoticons associated with each idiom response, enabling AIML to support and recognize multiple emotionally contextualised responses for each idiom. This innovative step in the AIML format enriches AI interactions, making them more nuanced and akin to human-like communication. It allows the AIML system to respond not just based on the textual content of user queries but also to interpret and reflect the underlying emotional tone of the conversation.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 01 January 2024

XX. SELECTION AND COMPILATION OF RESPONSES

In the compilation process of responses for the 3,500 idioms, each paired with 16 emoticons, systematic formatting is key. For every idiom, one of the 16 responses is earmarked as the default response, based on its relevance and the predominant emotion it represents in typical dialogues. Each idiom and its corresponding responses are then formatted into the AIML structure, incorporating the newly added 'emotion' attribute within AIML tags. This structured approach ensures that the AIML system can effectively process and utilise these responses in interactions.

XXI. IMPLEMENTING ADDITIONAL AIML TAGS

In addition to the 'emotion' attribute, other attributes like 'avatar' are also incorporated, further enhancing the richness of the responses. The 'avatar' attribute aligns an emoji or graphical representation with the emotional tone of the response, adding a visual dimension to the textual interaction.

XXII. EXAMPLE OF ENHANCED AIML STRUCTURING

For illustration, consider the idiom "He is the apple of my eye". Its AIML structure, accommodating multiple responses for various emotions, see Fig.1.

Fig.1 Sample AIML

In this AIML example, each `<li>` tag within the `<random>` tag contains a unique response associated with a specific emotion and avatar, allowing the AI system to choose a response that matches the emotional context of the user's input. Legacy AIML parsers will pick any of the statements at random and the authors' proprietary system picks the one marked `default="true"` where no input emotion has been matched to an emoticon.

XXIII. ENHANCING THE FLEXIBILITY AND RESPONSIVENESS OF AIML

Implementing Multi-Response Capability

The capability of AIML to accommodate multiple responses for a single idiom, enabled by the 'emotion' attribute, significantly enhances its flexibility and adaptiveness. This multi-response feature introduces a degree of dynamism previously unattainable, allowing the AI system to select from a variety of emotionally nuanced responses depending on the context of the conversation.

XXIV. ADDRESSING TECHNICAL IMPLEMENTATION CHALLENGES

Implementing these enhancements necessitates modifications to the AIML parser to recognize and process the new attributes, such as 'emotion' and 'avatar'. This task requires a profound understanding of AIML's architecture and meticulous coding to ensure seamless integration with existing AIML standards and functionality. Ensuring that the emotional responses are contextually appropriate is a critical challenge in this implementation. The AI system is tasked with accurately interpreting the user's emotional state and selecting a response that aligns with that emotion, requiring sophisticated algorithms capable of discerning intricate language and emotional nuances. Moving forward, this approach to enhancing AIML paves the way for more advanced stages of development, focusing on refining the system's ability to navigate and respond to the complex emotional landscape of human interactions. This progression towards a more emotionally intelligent AI holds significant implications for the future of AI and human-computer interaction.

XXV. EXTENDING PYAIML FOR EMOTION AND AVATAR TAGS

Enhancing PyAIML for New Attributes

As the authors' project advances, they delve into enhancing the PyAIML parser, a Python adaptation of AIML, to support the novel 'emotion' and 'avatar' attributes integrated into the AIML structure. This enhancement is pivotal for processing these innovative tags within AIML files, marking a significant step in the evolution of AI communication. The 'emotion' attribute empowers the PyAIML parser to grasp and reflect the emotional tone of user interactions, while the 'avatar' tag introduces a visual element to the AI's responses. These enhancements bring a dynamic and more human-like quality to the interactions facilitated by the AI system.

XXVI. IMPLEMENTING ADVANCED PATTERN MATCHING

A key aspect of enhancing PyAIML is the improvement of its pattern-matching capabilities through the integration of Regular Expressions (Regex). This enhancement enables the parser to adeptly handle and interpret complex and varied user inputs, accommodating natural language variations including colloquial expressions and unstructured phrases. Further, the enhancement for recognizing common words effectively is implemented, ensuring the AI system maintains

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 01 January 2024

coherent and contextually relevant dialogues across a wide range of conversational inputs.

XXVII. DATABASE INTEGRATION FOR ENHANCED FUNCTIONALITY

The next enhancement phase involves adapting PyAIML to function seamlessly with relational databases, such as MySQL or Microsoft SQL Server. This integration significantly uplifts the AIML system's functionality and scalability, especially considering the voluminous data comprising numerous idioms and their corresponding responses.

XXVIII. COMPILING AND STORING AIML IN A RELATIONAL DATABASE FORMAT

Saving Compiled AIML to Database

The next critical step involves saving the compiled AIML files, now rich with 'emotion' and 'avatar' attributes, into a relational database. This process transforms the AIML data into a format congruent with relational database structures, optimising data management and retrieval capabilities. The database structure is tailored to align with the AIML file format, encompassing categories, patterns, templates, and the newly integrated attributes. This organisation ensures coherent and accessible data storage.

XXIX. ENHANCING DATA RETRIEVAL AND MANAGEMENT

Storing AIML data in a relational database streamlines the retrieval and management of AI responses. The database facilitates swift access to specific idioms and their related responses, enhancing the AI's interaction efficiency. Robust data management tools provided by the database, including backup, recovery, and maintenance capabilities, ensure the long-term durability and integrity of the AIML data, a critical factor given the dynamic nature of AI interactions.

XXX. SYNCHRONISING PYAIML WITH DATABASE

The final enhancement involves synchronising PyAIML with the relational database. This synchronisation guarantees that updates in the AIML files are accurately reflected in the database, achieved through database connectors and synchronisation scripts within PyAIML. Moving forward, these enhancements in PyAIML set the stage for subsequent phases of the project, aiming to refine and optimise the AI's interactive abilities and emotional intelligence.

XXXI. DEVELOPING THE PYTHON FLASK API FOR AIML INTERACTION

Introduction to API Development with Python Flask

The next crucial step in the project involves creating an Application Programming Interface (API) using Python Flask. This API serves as a gateway for users to interact with the AIML mechanism developed with PyAIML and integrated with a MySQL database. Flask, a lightweight and powerful web framework for Python, is an ideal choice for this purpose due to its simplicity, flexibility, and ease of use.

XXXII. SETTING UP THE FLASK ENVIRONMENT

The first phase in developing the API involves setting up the Flask environment. This setup includes installing Flask using Python's package manager, pip, and creating a new Flask application instance. The structure of the Flask application is organised into various components, including routes, templates, and static files, facilitating a modular and maintainable codebase.

XXXIII. DESIGNING API ENDPOINTS

The core of the Flask application lies in designing API endpoints that users can access to interact with the AIML system. These endpoints are defined as routes in Flask, each corresponding to a specific functionality of the AIML mechanism. For instance, an endpoint might be created for submitting user queries to the AIML system and retrieving responses. Each route is associated with a Python function that handles the incoming HTTP requests, processes them using the PyAIML and MySQL setup, and returns the appropriate responses. Care is taken to ensure these functions are efficient and secure, providing a seamless and safe user experience.

XXXIV. INTEGRATING PyAIML AND MySQL WITH FLASK

The integration of PyAIML and MySQL with Flask application is a critical aspect of the API development. This integration allows the Flask application to interact with the AIML mechanism, processing user inputs and fetching responses from the MySQL database. To achieve this integration, Python scripts are written within the Flask application that connects to the MySQL database using appropriate connectors. These scripts utilise PyAIML functionalities to parse user inputs, match them with the AIML patterns stored in the database, and generate the relevant AI responses.

XXXV. IMPLEMENTING THE API AND ENSURING ROBUST FUNCTIONALITY

A vital feature of the Flask-based Gateway API is processing user inputs. When a user submits a query through the API, the Flask application receives this input and passes it to the PyAIML engine. The engine, in turn, processes the input, matching against the AIML patterns, and retrieves the corresponding response. Special attention is given to handling various types of user inputs, including those with different emotional contexts, as identified by the 'emotion'

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 11 Issue 01 January 2024

and 'avatar' tags in the AIML setup. The system is designed to understand and respond to these inputs accurately, reflecting the emotional intelligence of the AI.

Security is a paramount concern in API development. Measures are implemented to safeguard against common web vulnerabilities, such as SQL injection and cross-site scripting (XSS). Flask provides several built-in tools to help secure the application, which are utilised to their full potential. Scalability is another crucial factor considered in API development. The Flask application is designed to handle a significant number of concurrent user requests without compromising performance. This scalability is crucial for the widespread adoption and usability of the API.

While the backend functionality is critical, providing a user-friendly interface for interacting with the API is equally important. The Flask application includes templates that render user interfaces for inputting queries and displaying responses. These interfaces are designed to be intuitive and easy to use, catering to a wide range of users.

Before deployment, the Flask API undergoes thorough testing to ensure its functionality and robustness. This testing includes unit tests for individual components and integration tests for the entire application. Special attention is given to testing the API's interaction with the PyAIML engine and the MySQL database, ensuring that the entire system works seamlessly together. Once tested, the Flask API is deployed on a suitable web server. The deployment process involves configuring the server settings, setting up the necessary environment variables, and ensuring that the API is accessible to users. The creation of a Python Flask API for interacting with the AIML mechanism developed with PyAIML and MySQL represents a significant milestone in the project. The focus on user experience, security, scalability, and robust testing and deployment strategies ensures that the API is efficient, secure, and capable of handling diverse user interactions.

XXXVI. RESULTS AND ANALYSIS

The project's initial phase involved collecting 3,500 English idioms, presenting a significant opportunity to explore the processing of free-form text using Python and pandas. This process illustrated the efficiency of automation in data handling compared to manual methods. Through Python scripts, the authors swiftly crawled, extract, and structure idiomatic data.

A pivotal aspect of the project was leveraging the OpenAI GPT-3.5-turbo API to generate responses for idioms paired with 16 predefined emoticons from the Crocels list [1]. This approach significantly accelerated the creation of the dataset. The API's capability to process and understand the nuanced meanings of idioms in conjunction with emotional contexts, as represented by emoticons, was instrumental in constructing a diverse and rich dataset. This advancement underscored the utility of advanced AI models in handling complex linguistic tasks.

The AI model provided multiple response options for each idiom-emoticon pair, from which the authors selected suitable ones. This multi-response feature of the AI model was crucial in ensuring that the chosen responses accurately reflected the intended emotional tone and context. It demonstrated the model's ability to offer varied and contextually relevant responses, enhancing the quality and reliability of the dataset.

Utilising Natural Language Processing (NLP) methodologies, the authors efficiently validate the appropriateness of the responses generated by the AI model. This approach was significantly timesaving compared to manual review. NLP tools analysed the sentiment and contextual relevance of the responses, ensuring suitability. This automated validation process underscored the importance and effectiveness of NLP in streamlining linguistic data analysis.

The incorporation of a Gateway API and a relational database played a crucial role in system testing and access. This setup allowed for testing of the AIML parser and the integration of the idioms and emoticons database. The database provided a structured platform for storing and retrieving the AI responses, while the Gateway API facilitated seamless interaction with the AIML parser. This architecture was key in evaluating the system's performance and ensuring its functionality.

XXXVII. DISCUSSION

As the authors culminated the project, it is pertinent to reflect on the strides made and the future implications of this endeavour. This project, ambitious in its scope and meticulous in its execution, has significantly advanced the capability of Artificial Intelligence in comprehending and employing idiomatic English. The completion of this venture marks a noteworthy milestone in the journey towards creating AI systems that can interact with a level of understanding and emotional intelligence akin to humans. The collection and processing of over 3,500 English idioms, each paired with emotionally resonant emoticons, stand as a testament to the project's success. The use of Python for data handling, the application of the OpenAI GPT-3.5-turbo API for generating responses, and the integration of Natural Language Processing for response validation have collectively demonstrated the power of technology in enhancing AI's linguistic capabilities. The ability of the AI system to not just recognize idioms but also to understand their contextual and emotional nuances is an achievement that bridges a significant gap in AI-human interaction.

REFERENCES

[1] J. Bishop (2019). Assisting human interaction (United States Patent) [Review of Assisting human interaction]. https://patents.google.com/patent/US10467916B2

[2] J. Bishop (2015). Supporting communication between people with social orientation impairments using affective computing technologies: Rethinking the autism spectrum. In Assistive technologies for physical and cognitive disabilities (pp. 42-55). IGI Global: Hershey, PA.

[3] J. Bishop, J., & M. Reddy (2003). The role of the Internet for educating individuals with social orientation impairments. *Journal of Computer Assisted Learning*, *19*(4), 546-556.

[4] P.G. Kirchschläger (2021). Digital transformation and ethics: ethical considerations on the robotization and automation of society and the economy and the use of artificial intelligence. Nomos Verlag.