UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Telecommunications Engineering

Functional design of a bot for query redirection in an employee web page oriented.

End of Degree Project

Bachelor's Degree in Telecommunication Technologies and Services Engineering

AUTHOR: Bamhaoued, Karim

Tutor: González Ladrón de Guevara, Fernando Raimundo

ACADEMIC YEAR: 2022/2023

Resumen

Esta tesis explora el desarrollo e implementación de un chatbot de inteligencia artificial diseñado para manejar consultas de empleados y recuperar información de una base de datos MySQL. Al examinar la importancia de la gestión de recursos humanos, la evolución de los chatbots y su uso en recursos humanos, el proyecto tiene como objetivo crear un chatbot que aborde las necesidades de la organización y brinde soporte a los empleados. La tesis profundiza en las técnicas y metodologías de construcción de chatbots, demuestra el despliegue y la funcionalidad, y discute los resultados obtenidos, las limitaciones y las posibles líneas de investigación futuras. La exitosa integración de PyMySQL, un modelo de aprendizaje profundo y técnicas de procesamiento del lenguaje natural resulta en un chatbot que mejora la eficiencia, el ahorro de costos y la satisfacción de los empleados.

Resum

Esta tesi explora el desenvolupament i la implementació d'un xatbot d'intel·ligència artificial dissenyat per a gestionar consultes dels empleats i recuperar informació d'una base de dades MySQL. Examinant la importància de la gestió de recursos humans, l'evolució dels xatbots i el seu ús en recursos humans, el projecte té com a objectiu crear un xatbot que aborde les necessitats de l'organització i done suport als empleats. La tesi aprofundeix en les tècniques i metodologies de construcció de xatbots, demostra el desplegament i la funcionalitat, i discuteix els resultats obtinguts, les limitacions i les possibles línies de recerca futures. La integració exitosa de PyMySQL, un model d'aprenentatge profund i tècniques de processament del llenguatge natural resulta en un xatbot que millora l'eficiència, l'estalvi de costos i la satisfacció dels empleats.

Abstract

This thesis explores the development and implementation of an AI chatbot designed to handle employee queries and retrieve information from a MySQL database. By examining the importance of human resource management, the evolution of chatbots, and their use in HR, the project aims to create a chatbot that addresses organizational needs and supports employees. The thesis delves into chatbot-building techniques and methodologies, demonstrates deployment and functionality, and discusses the achieved results, limitations, and potential future research directions. The successful integration of PyMySQL, a deep learning model, and natural language processing techniques results in a chatbot that improves efficiency, cost savings, and employee satisfaction.

Index:

1. Introduction

1.1. Motivation

In today's increasingly digital world, businesses must continually adapt to new technologies to stay competitive. An essential aspect of this adaptation is the development of efficient and effective employee web pages. However, despite significant investments in web page design and development, employees still struggle to find the information they need quickly.

Designing and implementing a bot for query redirection in an employee web page can solve this problem. The bot can analyze employee queries and redirect them to the appropriate web page or resource, reducing the time and effort employees spend navigating the web page. This will increase employee satisfaction, reduce frustration, and ultimately boost productivity.

1.2. Objectives

The proposal of this project is to introduce an AI-powered chatbot into the HRM department to enhance the employee experience, automate administrative tasks, and provide real-time decision-making capabilities. By integrating AI technology, businesses can streamline HR processes, increase efficiency, and reduce operating costs. The chatbot will be customized to handle common HR queries, such as vacation policies, benefits information, and performance evaluations, freeing up HR professionals to focus on more complex tasks, such as managing employee relations and developing talent. The implementation of AI-powered chatbots in HRM presents an exciting opportunity to revolutionize business operations and create a more engaging work environment.

1.3. Structure of the thesis

This document has been organized in the following format:

1. **Introduction:**

In this chapter, the author presents the context for the project, followed by their personal motivation for pursuing this research. Finally, the chapter outlines the objectives of the project.

2. **Human resources management:**

this chapter will provide an overview of the importance of human resource management and the challenges faced by HR professionals. It will review the literature on HR management, describe the research methodology used, and present the findings of the study, which will include data on current HR practices, challenges and effectiveness of HR practices.

3. **History, evolution, architecture and different technologies of AI chatbots:**

this section will discuss the history and evolution of chatbots, their architecture, and different technologies used in their development.

4. **AI Chatbots in Human Resource Management:**

The purpose of this chapter is to give an overview of the problem, goals to achieve through the bot implementation, design constraints, and user issues to address. This allows us to design a bot tailored to the organization's needs and provide effective support for employees in their work

5. **Building chatbots:**

We explore the various techniques and methodologies involved in creating chatbots, focusing on the essential components, tools, and best practices for effective development. This comprehensive guide will enable readers to understand the intricacies of chatbot construction, ensuring the development of chatbots that meet the needs of users and enhance their overall experience.

6. **Deploying the chatbot:**

In this chapter, we'll demonstrate the chatbot's deployment and functionality via a web interface, enabling users to interact with it and observe real-time responses. The goal is to showcase the chatbot's practical use and its capability to comprehend and effectively answer user queries.

7. **Conclusions and future lines of research:**

This project aimed to develop a chatbot capable of managing employee queries and efficiently retrieving information from a MySQL database. By integrating PyMySQL, a deep learning model, and natural language processing techniques, the chatbot successfully addresses these objectives, resulting in increased efficiency, cost savings, and improved employee satisfaction. Despite some limitations, potential future research could further enhance its performance and user experience.

8. **Bibliographic references**

2. Human resource Management:

2.1 Fundamentals of Human Resource Management

HRM is a strategic, integrated and coherent approach to the employment, development and well-being of the people working in organizations. The purpose of HRM is to enable employees to contribute effectively and productively to the overall company direction and the accomplishment of the organization's goals and objectives, while at the same time enabling employees to achieve their own goals and objectives. HRM is concerned with the acquisition, development and reward of employees, as well as the management of change within the organization (Armstrong, 2017, p. [page 3]).

By defining HRM as a strategic approach to managing the employment, development, and well-being of people within organizations, it is important to understand the functions of HRM that contribute to achieving this goal. One key area of HRM is employee relations, which involves managing the relationship between the employer and employees to create a positive work environment and maintain effective communication. Employee relations encompass a variety of activities, including employee engagement, conflict resolution, and managing employee benefits.

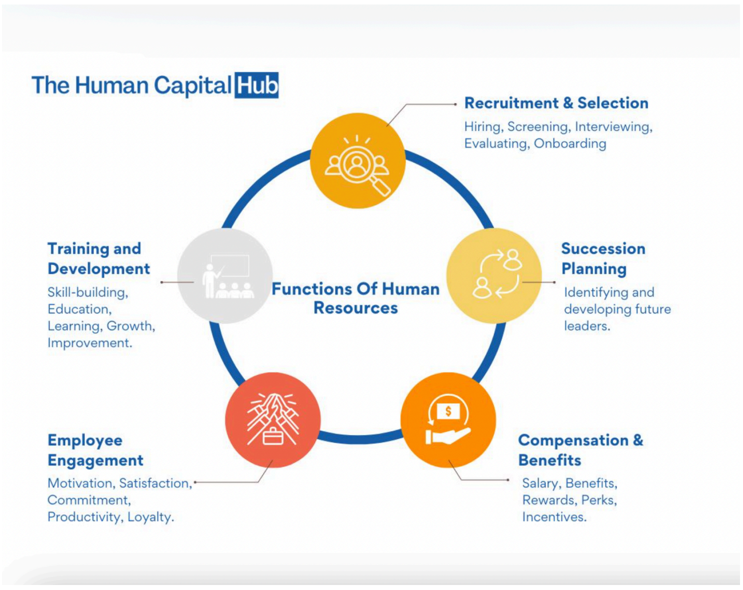

The Human Capital Hub
Recruitment & Selection
Hiring, Screening, Interviewing, Evaluating, Onboarding
Succession Planning
Identifying and developing future leaders.
Compensation & Benefits
Salary, Benefits, Rewards, Perks, Incentives.
Employee Engagement
Motivation, Satisfaction, Commitment, Productivity, Loyalty.
Training and Development
Skill-building, Education, Learning, Growth, Improvement.
Functions Of Human Resources

Figure 1: Functions of Human Resources

Human resource professionals play a critical role in managing an organization's human capital. They ensure the development and implementation of effective employee benefit systems, training and development programs, and performance evaluation and reward systems. Additionally, they work to maintain positive employee relations by fostering a harmonious relationship between management and workers and ensuring compliance with labor laws and regulations.

In today's business landscape, organizations are facing intense competition, leading them to adopt new technological advancements to remain competitive in the market. (Narayanan, 1998). In this scenario, man–machine interaction has become much more crucial. The industries can run smoothly and achieve the business objectives by the collaboration between human and computers. Business can be effective in the different functional domains by using different technical components (Bos et al., 2019a).

As organizations face intense competition in today's business landscape, the integration of technology and human resources has become crucial for their success. The use of HRIS, a software application that merges HR management and information technology, allows companies to streamline their workforce management processes, making them more efficient and effective. By automating HR processes, such as recruiting, onboarding, performance management, and employee records management, HRIS can free up HR staff to focus on strategic initiatives, while also reducing administrative errors and costs (SelectHub. ,2021).

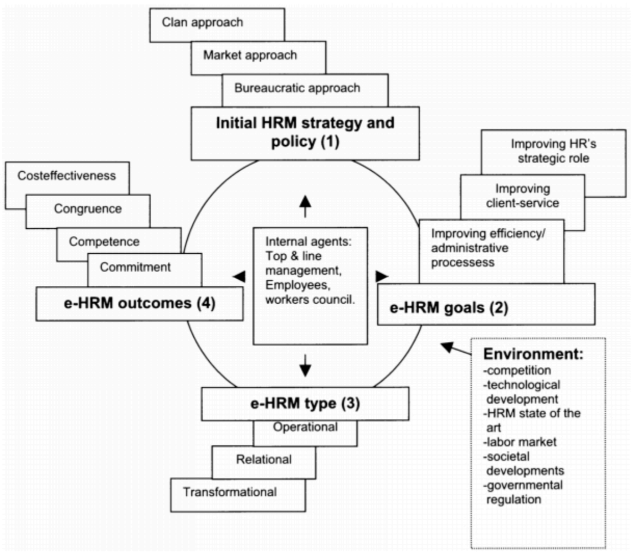

Clan approach
Market approach
Bureaucratic approach
Initial HRM strategy and policy (1)
Improving HR's strategic role
Improving client-service
Improving efficiency/ administrative processess
Costeffectiveness
Congruence
Competence
Commitment
e-HRM outcomes (4)
Internal agents: Top & line management, Employees, workers council.
e-HRM goals (2)
Environment: -competition -technological development -HRM state of the art -labor market -societal developments -governmental regulation
e-HRM type (3)
Operational
Relational
Transformational

Figure 2: e-HRM model

2.2 Human Resource Information Systems (HRIS)

HRIS is a system used to acquire, store, manipulate, analyze, retrieve, and distribute pertinent information about an organization's human resources. The system combines hardware, software, and data and provides information for decision making, communication, and administrative activities (Kavanagh and Johnson, 2018, p. 5).

In other words, It's used to acquire and store information about employees, such as personal information, job details, and performance data. The system can manipulate and analyze this information, providing insights and trends that can help decision-makers better understand their workforce. Additionally, HRIS can retrieve and distribute this information as needed, such as generating reports or providing data to other systems.

2.2.1 Features of an Effective HRIS

Each HRIS has a number of "essential features" that cover other important processes and services. Most of them are simply marketing tactics. It is therefore essential to be careful when choosing an HRIS (Features of an Effective HRIS. IceHrm, 2021),One of the truly essential features of an effective HRIS is integration with other systems.

A well-implemented HRIS should be able to integrate with other HR and organizational systems such as payroll, benefits, performance management, and accounting to ensure that all employee data is accurately and efficiently collected, analyzed, and reported (Noe et al., 2017, p. 239).

Flexibility is another important feature of an effective HRIS. HRIS should be flexible enough to adapt to the changing needs of the organization. This includes the ability to customize workflows, forms, and reports to fit the unique needs of the organization. (Kavanagh et al., 2019, p. 106)

For example, an HRIS should be able to accommodate changes in job titles, job descriptions, and other HR policies without requiring extensive programming or customization. This flexibility allows the HRIS to evolve as the organization changes and grows, ensuring that it remains a useful tool for managing human resources. Additionally, HRIS should be able to handle multiple languages and currencies, as many organizations have a global presence and require HRIS that can operate across borders.

Data accuracy is an important feature of an effective HRIS. The system should have accurate and up-to-date information on employees, such as personal information, job history, and performance data. This helps to ensure that HR decisions are based on reliable information, and that employees receive the appropriate benefits and compensation based on their qualifications and experience. Inaccurate or outdated information can lead to errors in decision making and negatively impact employee satisfaction and performance. HRIS should also have mechanisms in place to maintain

data accuracy, such as regular data audits and user access controls to prevent unauthorized changes to employee information.

In the healthcare industry, accurate and timely information is crucial for providing quality patient care. A study conducted by the Agency for Healthcare Research and Quality (AHRQ) found that inaccurate patient information, including incomplete medical histories and incorrect medication orders, led to medication errors and adverse events in hospitals (AHRQ, 2020). HRIS can play a significant role in ensuring data accuracy for healthcare organizations by providing up-to-date employee information to support scheduling, staffing, and training decisions. This helps to ensure that healthcare providers have the necessary resources to deliver quality care to patients.

Another feature is Self-service, refers to the ability of employees to access and manage their own personal HR-related information, such as benefits enrollment, paid time off requests, and updating personal information. Self-service features provide employees with greater control and convenience, while also reducing the administrative burden on HR staff.

Self-service applications provide employees, managers, and HR professionals with direct access to data and HR transactions. Through self-service, employees can enter and update their own information, view and modify their benefits and compensation, and access training and development programs. Managers can use self-service to view and approve employee requests, conduct performance evaluations, and make staffing decisions. HR professionals can use self-service to manage recruiting and onboarding processes, administer employee benefits and compensation, and maintain employee records. Self-service applications can help to reduce administrative costs, improve data accuracy an consistency, and increase employee and manager satisfaction (Kavanagh, Thite & Johnson, 2019, p. 287).

Reporting and analytics are important features of HRIS that allow organizations to make informed decisions based on data analysis. HRIS can generate reports on employee data, such as headcount, turnover rate, compensation and benefits, and performance metrics. These reports can help identify trends and areas for improvement, allowing organizations to make data-driven decisions that improve the overall performance and effectiveness of the HR function.

Moreover, HRIS can also offer data visualization tools that help HR professionals create visual representations of complex data sets, making it easier to identify patterns and insights. Advanced analytics tools, such as predictive analytics and machine learning algorithms, can also be integrated into HRIS to forecast future trends and outcomes, and to automate routine HR tasks.

According to Kavanagh, Thite, and Johnson (2019), "Reporting and analytics capabilities are critical for HR functions to demonstrate their value and impact to the organization. These capabilities enable HR professionals to quantify the effectiveness of HR policies and programs, and to provide meaningful insights to decision makers at all levels of the organization" (p. 294).

Security is a critical feature of any HRIS as it deals with sensitive employee data such as social security numbers, salary information, and personal identification. A breach of this information can have severe consequences, including legal repercussions and reputational damage. It is, therefore, crucial for an HRIS to have robust security measures in place to protect this data. This includes access controls to limit who can view and modify employee information, data encryption to prevent unauthorized access, and regular security audits to identify potential vulnerabilities.

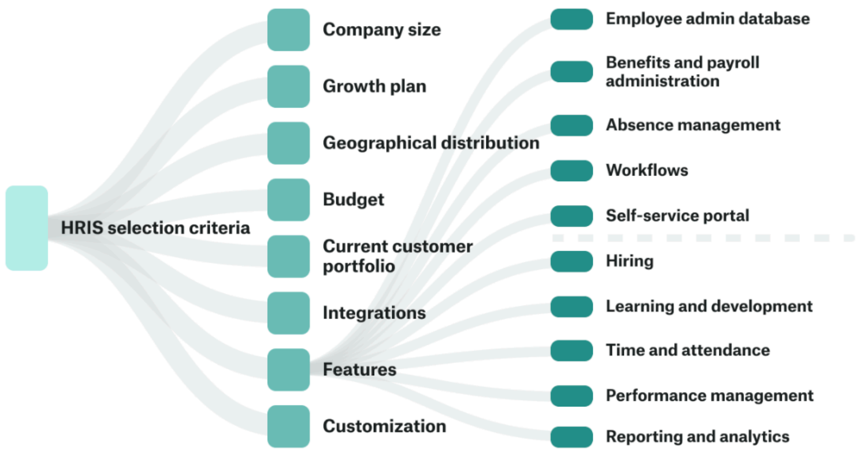


Figure 3: HRIS selection criteria diagram

2.3 Employee Portals

In addition to the core features listed above, many modern HRIS systems include employee portals as a key component.

Technically speaking, an employee portal offers a browser-based user interface providing access to personalized information, resources, and applications. In many cases, an employee portal is the primary tool through which employees do their work. Ideally, employee portals yield organizations and employees different benefits, such as reducing information overload, lowering organizational costs, improving corporate communication and knowledge management (KM), as well as enhancing employee productivity (Tojib et al., 2006).

Today, many companies, especially large ones, offer their employees a portal. A 2006 US study by Forrester Research, Inc. indicated that 46% of large companies ran an employee portal, and another quarter planned to establish one by 2008 (For- rester, 2006).

The use of employee portals has been growing steadily and, despite many companies restricted IT budgets, investments in portal solutions are still growing. However, portal projects are usually complex, time and cost-consuming with a high failure risk (Remus, 2006). And although IT departments and decision-makers have to justify portal investments, a significant number of companies do not assess their portal implementations actual benefits (Brown et al., 2007). Companies that do so, often use monetary indicators and cost-benefit analysis methods (White, 2003).

These success-measurement approaches do not, however, take intangible impacts and intervening environmental variables into account. A portal's success cannot, however, be measured by just its reach, and practitioners should not simply rely on "hit counts" as measures of success (Damsgaard and Scheepers, 1999). Clearly, a comprehensive measurement of portal success would also need to consider a portal's intangible effects to detect areas of potential improvements and justify present and future investments in portal solutions.

According to the hypotheses of the paper "An Empirical Investigation of Employee Portal Success" by Benbya et al. (2004), the perceived process quality and collaboration quality of an employee portal can have a significant impact on user satisfaction and usage of the portal. Specifically, the higher the perceived process quality, the more satisfied users are with the portal and the more they will use it. Similarly, the higher the perceived collaboration quality, the more satisfied users are with the portal and the more they will use it.

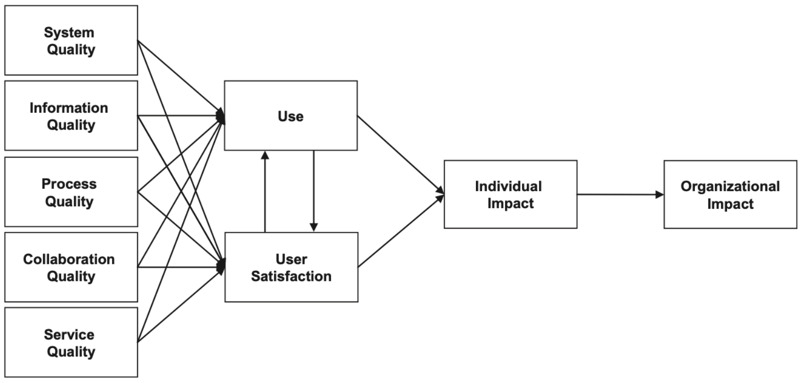


Figure 4: Employee portal quality model

In this regard, AI chatbot can play a crucial role in enhancing collaboration quality within an employee portal. By providing quick and accurate responses to employee queries and concerns, chatbots can help reduce response times and increase accessibility to important information. This can lead to improved communication and collaboration between employees and HR departments, as well as between employees themselves.

3. History, evolution, architecture and different technologies of AI chatbots

3.1 A brief history of chatbots :

Chatbots have a long history that dates back to the mid-20th century when computer scientists first began exploring natural language processing (NLP) and machine learning algorithms . One of the earliest chatbots was ELIZA, developed by Joseph Weizenbaum in 1966, which used pattern matching and substitution techniques to simulate conversation (Weizenbaum, 1966).

ELIZA's success paved the way for other chatbots, such as PARRY and Jabberwacky, to be developed in the 1980s and 1990s. However, it wasn't until the mid-2000s, with the rise of social media platforms like Facebook and Twitter, that chatbots began to gain widespread popularity.

For several decades, chatbots largely followed the rule-based approach used by ELIZA with minor improvements such as speech synthesis and emotion management. However, in 2001, a conversational agent called SmarterChild was introduced by ActiveBuddy, Inc. (now Colloquis), which operated on AOL Instant Messenger and MSN Messenger. SmarterChild was designed to provide quick access to news, weather forecasts, sports results, and other information by connecting to a knowledge base. This was an innovative approach, inspired by the rise of instant messaging platforms such as SMS. However, due to the limitations of natural language processing technology at the time, chatbots on these platforms were eventually forgotten by history.

The next significant advancement for conversational agents came from the IBM Watson AI project, which had been under development since 2006. The primary goal of the project was to design an agent capable of winning the American TV show Jeopardy! In 2011, the Watson agent successfully defeated two of the show's former champions. Jeopardy! is an interesting challenge for NLP systems because the questions often involve wordplay and require fast information retrieval from vast knowledge bases. However, the Watson AI, in its previous form, could only provide one-line answers and was incapable of engaging in a proper conversation with a human user.

The 2010s saw the emergence of virtual assistants, including Siri, Cortana, Google Assistant, and Alexa, among others. These agents introduced the concept of goal-oriented dialog and conversation to the field of chatbots. Additionally, the Messenger

Platform for Facebook Messenger was released in 2016, which enabled the creation of conversational agents for non-AI related companies.

Since the release of the Messenger Platform, the field of chatbots has continued to evolve rapidly. Many companies have started using chatbots for customer service, sales, and marketing, as they can provide 24/7 support and handle a large volume of inquiries. Moreover, advancements in machine learning and natural language processing have made it possible for chatbots to understand and respond to more complex queries.

In recent years, there has been a growing trend towards using chatbots in healthcare, mental health, and therapy. Chatbots have been developed to provide support for individuals with mental health issues, provide health-related advice and information, and even offer therapy sessions.

As the field of chatbots continues to evolve, it is likely that they will become even more sophisticated and integrated into our daily lives. With advancements in technologies such as voice recognition and computer vision, it is possible that chatbots will soon be able to understand and respond to non-verbal cues, such as facial expressions and body language. As such, chatbots will continue to play an important role in shaping the future of human-computer interaction.

According to Abdul-Kader, S. A., & Woods, J. C. (2015), the evolution of chatbots can be categorized into three main stages:

1. First Generation: These chatbots are rule-based and use predefined templates to respond to user input.
2. Second Generation: These chatbots use machine learning algorithms to improve their responses over time. They can learn from user interactions and adapt to new situations.

3. Third Generation: These chatbots use advanced AI technologies, such as natural language processing and deep learning, to provide more human-like conversation experiences. They can understand context and perform complex tasks.

3.2 Architecture of chatbots:

A chatbot is an integration of several components working together towards a common objective. These components are interconnected and their relationships are illustrated in Figure 5, providing a visual representation of a conversational agent's structure.

When a user sends a message to a chatbot designed to solve employee queries, the chatbot goes through a series of components to determine the appropriate response. The first component is the language identification module, which processes the message using various methods to identify the language being used. The message is then passed on to the intent classifier module, which uses machine learning algorithms to determine the user's intent based on the message received.

Using the metadata from the message and the inferred intent, the chatbot's backend accesses additional information to determine the appropriate response. This information includes previous conversation messages and other relevant data. Based on this information, the chatbot determines an appropriate action or sequence of actions to perform. For example, the chatbot may ask clarifying questions if the intent is unclear, or it may reactivate a user account if that is the user's intention.

Finally, the action handler module executes the determined action, taking into account the environment in which the chatbot is operating. This allows for the same action to be executed in different ways, depending on the platform or website where the chatbot is operating.

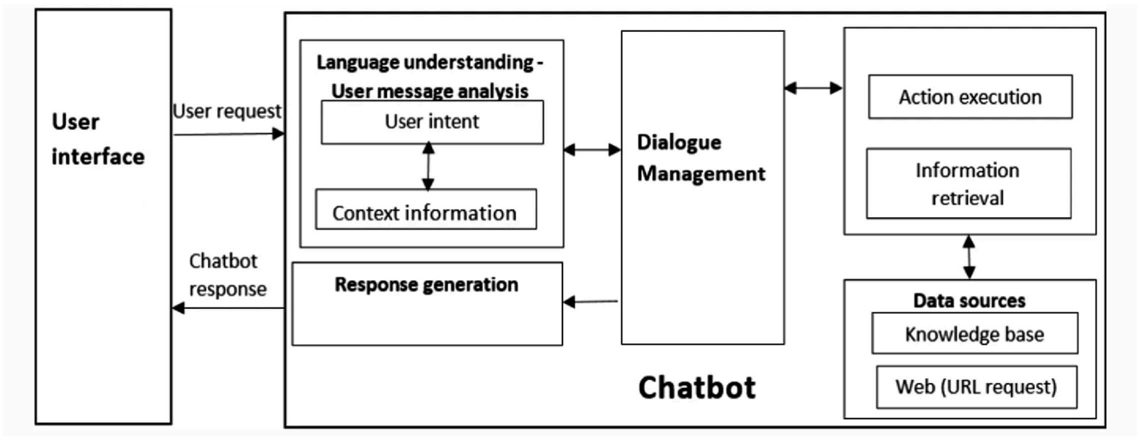


Figure 5 : conceptual diagram of a chatbot

3.2.1 Rule-based

Rule-based approaches involve defining a set of rules or heuristics that guide the chatbot's behavior. These rules are typically defined by domain experts or developers based on their understanding of the problem domain and the common patterns and questions that users might have.

One example of a rule-based approach is a decision tree, which involves branching based on a set of predefined rules or criteria. Another example is a template-based approach, where the chatbot selects a pre-defined response template based on the user's input.

Other rule-based models include hand-crafted state machines, regular expression matching, and pattern recognition techniques. These approaches require manual intervention and domain expertise to design and refine the rules.

Among the various rule-based models, the decision tree approach has been found to be particularly effective in employee-based portals due to its ability to handle complex decision-making processes.

3.2.2 Decision Tree Method

The decision tree model can also be applied to employee portals to provide effective support for employee queries. By defining a set of rules and criteria based on the nature of employee queries, the decision tree can help classify and predict the appropriate responses. The model can be used to help employees understand company policies, provide information on benefits, and answer questions related to performance evaluations. With the ability to customize the decision tree based on the needs of the organization, this approach can be highly effective in providing accurate and consistent responses to employee queries.

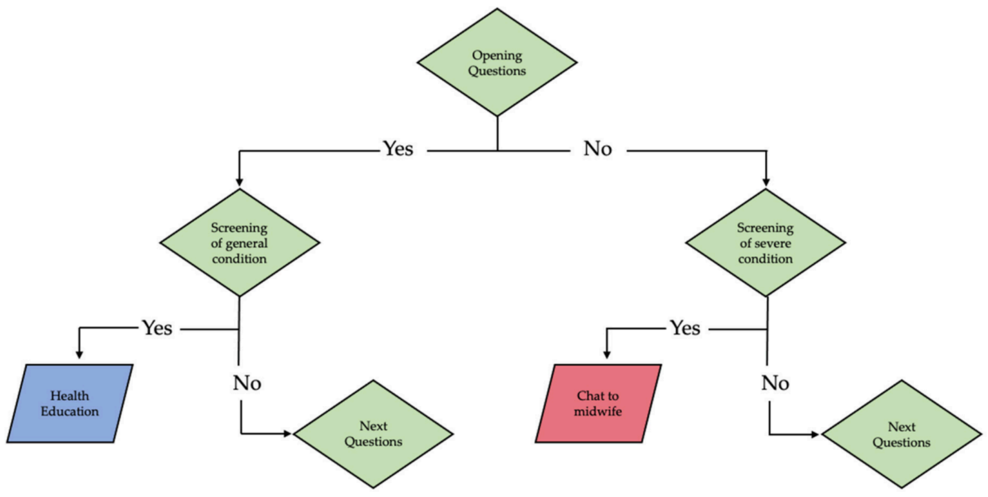


Figure 6 : An illustration of the hierarchical rules of a decision tree

In Figure 6, the tree is a hierarchical rule directed from top to bottom, starting with a decision node (green diamond shape) as the root or initial decision. Then, each management strategy is followed by yes or no choices, representing certain events. Finally, the endpoints of decision trees are characterized by a leaf as a terminal node (parallelogram shape) consisting of blue for the education needed on general conditions and red as the option of talking to a midwife, representing severe conditions at the end of the tree. The outcome measures are generally attached to these endpoint.

3.2.3 Transformer (machine learning model)

Transformer-based models are a type of neural network architecture that was introduced in the paper "Attention is All You Need" by Vaswani et al. (2017). They have revolutionized the field of natural language processing (NLP) and achieved state-of-the-art performance on a wide range of NLP tasks, including machine translation, sentiment analysis, and language modeling.

The key innovation of transformer-based models is the attention mechanism, which allows the model to focus on different parts of the input sequence during processing. This attention mechanism replaces the recurrent and convolutional layers that were commonly used in previous NLP models.

The transformer model uses a multi-head attention mechanism to capture dependencies between different parts of the input sequence, which allows it to model long-range dependencies more effectively than other neural network architectures. This makes it well-suited for tasks that involve understanding the context of a sequence of text, such as natural language understanding and text classification.

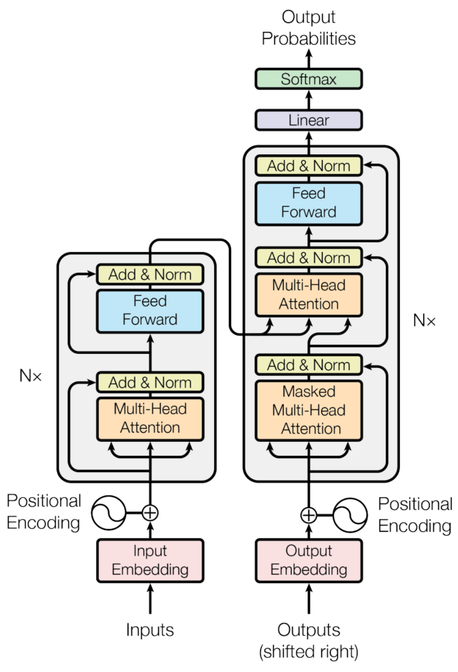


Figure 7: The Transformer - model architecture

BERT (Bidirectional Encoder Representations from Transformers):

BERT is a pre-trained language model that uses an unsupervised approach of language modeling on a large dataset to understand the context of input sentences. After pre-training, BERT can be fine-tuned on a task-specific supervised dataset to obtain good results.

There are two strategies for applying pre-trained models: fine-tuning and feature-based. Elmo uses a feature-based model where the model architecture is task-specific, and each task will use different pre-trained models for language representations.

In contrast, BERT uses a fine-tuning approach that utilizes bidirectional transformers encoders to understand language. BERT can understand the full context of a word by analyzing the terms before and after the word and finding the relationship between them.

Unlike other language models such as Glove2Vec and Word2Vec, which build context-free word embeddings, BERT provides context, making it a powerful tool for natural language processing tasks, including handling employee queries.

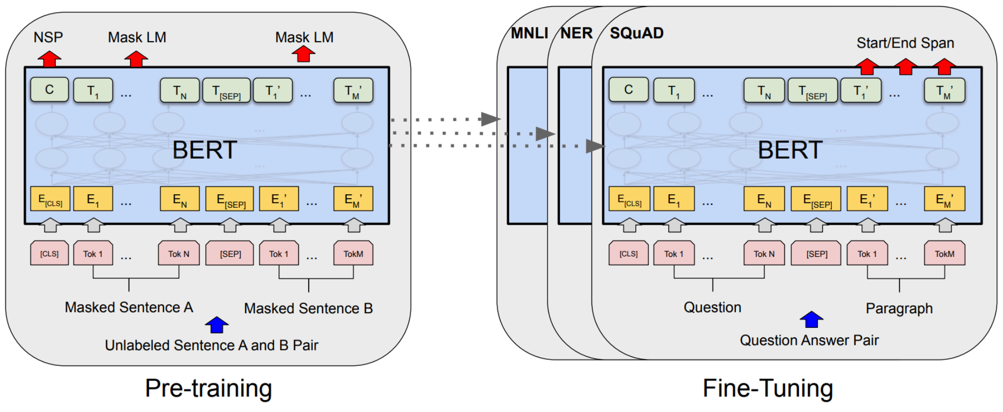


Figure 8 : pre-training and fine-tuning procedures for BERT

During the pre-training stage, a large amount of data is used to train the model, allowing it to learn about the relationships between words and the ways in which sentences are constructed. This knowledge is saved as parameters that can be inherited by other models. During the fine-tuning stage, the pre-trained model is re-trained for a new task. Because the model has already learned a great deal about language from the pre-training stage, it can often achieve good results with only a limited amount of new data.

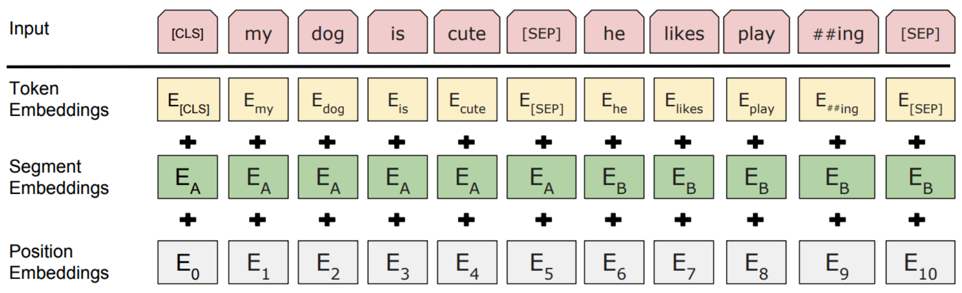

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
E[CLS] E_my E_dog E_is E_cute E[SEP] E_he E_likes E_play E_##ing E[SEP]
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_10

Figure 9 :BERT input representation

BERT requires the entire input to be fed as a single sequence, and to understand the input properly, it uses special tokens such as [CLS] and [SEP]. The [SEP] token should be inserted at the end of a single input. In cases where a task involves more than one input, such as NLI and Q-A tasks, [SEP] token helps the model to understand the end of one input and the start of another input in the same sequence input. The [CLS] token is a special classification token, and the last hidden state of BERT corresponding to this token (h[CLS]) is used for classification tasks.

To embed tokens, BERT uses Wordpiece embeddings input. In addition to token embeddings, BERT also employs positional embeddings and segment embeddings for each token. Positional embeddings contain information about the position of tokens in the sequence, while segment embeddings are useful when the model input has sentence pairs. Specifically, tokens of the first sentence are given a pre-defined embedding of 0, whereas tokens of the second sentence receive an embedding of 1 as segment embeddings.

The final input representation for BERT is the sum of the token, positional, and segment embeddings, which is then passed through deep bidirectional layers to generate output. The output is a hidden state vector of a predefined size for each token in the input sequence. These hidden states, obtained from the last layer of BERT, are utilized for various NLP tasks.

Pre-training and Fine-tuning : BERT was pre-trained on unsupervised Wikipedia and Bookcorpus datasets using language modeling. Two tasks namely Masked Language Model (MLM) and Next Sentence Prediction (NSP) were performed. During MLM, 15% of the tokens from the sequence were masked and then correct tokens were predicted at the final hidden state. To capture the relationship between sentence pairs given as input, NSP is used. For NSP, 50% of the data is labeled as isNext where sentence B of the input sequence is just the next sentence of sentence A from the dataset corpus. Another 50% of data is labeled as notNext where sentence B is not next to sentence A but any random sentence from the corpus dataset. Output hidden state corresponding to [CLS] token is used to predict the correct label and compute loss. After pre-training, BERT can be fine-tuned on the specific task-based dataset.

XLNet (Generalized Auto-Regressive model for NLU):

XLNet is a generalized AR pretraining method that uses a permutation language modeling objective to combine the advantages of AR and AE methods. The neural architecture of XLNet is developed to work seamlessly with the AR objective, including integrating Transformer-XL and the careful design of the two-stream attention mechanism. XLNet achieves substantial improvement over previous pretraining objectives on various tasks (Yang et al. (2019)).

Comparing the conventional AR language modeling and BERT for language pretraining. Given a text sequence $x = [x_1, \dots, x_T]$, AR language modeling performs pretraining by maximizing the likelihood under the forward autoregressive factorization:

$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x')\right)}, \qquad (1)$$

where $h_\theta(x_{1:t-1})$ is a context representation produced by neural models, such as RNNs or Transformers, and $e(x)$ denotes the embedding of x. In comparison, BERT is based on denoising auto-encoding. Specifically, for a text sequence x, BERT first constructs a corrupted version $\hat{\mathbf{x}}$ by randomly setting a portion (e.g. 15%) of tokens in x to a special symbol [MASK]. Let the masked tokens be $\bar{\mathbf{x}}$. The training objective is to reconstruct $\bar{\mathbf{x}}$ from $\hat{\mathbf{x}}$:

$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}}) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x')\right)}, \qquad (2)$$

where $m_t = 1$ indicates x_t is masked, and H_θ is a Transformer that maps a length-T text sequence x into a sequence of hidden vectors $H_\theta(x)$= $[H_\theta(x)_1, H_\theta(x)_2, \dots, H_\theta(x)_T]$. The pros and cons of the two pretraining objectives are compared in the following aspects:

- **Independence Assumption**: As emphasized by the ≈ sign in Eq. (2), BERT factorizes the joint conditional probability $p(\bar{\mathbf{x}} \mid \hat{\mathbf{x}})$ based on an independence assumption that all masked tokens $\bar{\mathbf{x}}$ are separately reconstructed. In comparison, the AR language modeling objective (1) factorizes $p_\theta(x)$ using the product rule that holds universally without such an independence assumption.

- **Input noise:** The input to BERT contains artificial symbols like [MASK] that never occur in downstream tasks, which creates a pretrain-finetune discrepancy. Replacing [MASK] with original tokens as in [10] does not solve the problem because original tokens can be only used with a small probability — otherwise Eq. (2) will be trivial to optimize. In comparison, AR language modeling does not rely on any input corruption and does not suffer from this issue.

- **Context dependency:** The AR representation $h_\theta(x_{1:t-1})$ is only conditioned on the tokens up to position t (i.e. tokens to the left), while the BERT representation $H_\theta(x)_t$has access to the contextual information on both sides. As a result, the BERT objective allows the model to be pretrained to better capture bidirectional context.

3.3. Language identification

Identifying the language of a text is often the initial and necessary step in a larger natural language processing pipeline. However, some languages may have homographs, such as "bank" in English and Dutch, which can cause confusion in algorithms due to the different meanings of the word in each language. Thus, it becomes essential to determine the correct language of a given text before further processing it. While this project assumes that messages are written in a single language, there are also instances where the problem involves detecting multiple languages within a single piece of text.

3.4. Intent classification

When a new message is received, the conversational agent needs to identify the user's intention or goal, which is typically modeled as a multiclassification problem. The labels are the possible names of user intentions, and various techniques can be used to solve this problem, ranging from simple keyword extraction to Bayesian inference that uses multiple messages to determine the user's request.

3.5. Knowledge management

The field of knowledge engineering has advanced significantly in enabling computers to handle knowledge, particularly in the 1980s. Early knowledge engineering techniques involved using an inference engine to manipulate facts and derive new knowledge using first and second-order logic. These techniques allow for generating answers to incomplete questions and can be easily translated into API calls. In the context of conversational agents, knowledge engineering can be highly beneficial for answering basic questions about general facts. For instance, digital assistants such as Siri and Amazon Alexa use internal knowledge inference methods to retrieve facts from the web and other sources. When asked about the current weather, for example, the assistant may use inference to generate a response based on the user's location and current weather conditions.

3.6. Responses generation

For a conversational agent to effectively communicate, it must possess the ability to generate coherent replies that align with the conversation's context. To achieve this, two modules typically work in tandem: one that generates a list of candidate replies and another that selects the most appropriate response or ranks them based on a specific metric. Two popular approaches have emerged in addressing this subproblem: retrieval-based and generative-based methods.

Retrieval-based techniques rely on a large database of pre-existing responses and match them with information from the user's message to find the most appropriate answer.

This information can be a simple regular expression that searches for particular sentence structures or the output of a machine learning model. The primary advantage of this approach is that the chatbot's maintainers can control every answer, thus ensuring that inappropriate replies are avoided.

In contrast, generative-based methods use machine learning models, such as neural networks, to generate responses on the fly based on the conversation's context. This approach allows for more flexibility and creativity in responses, but it also poses a challenge of ensuring that the responses remain coherent and appropriate. Generative-based methods also require a large dataset for training and can be computationally intensive.

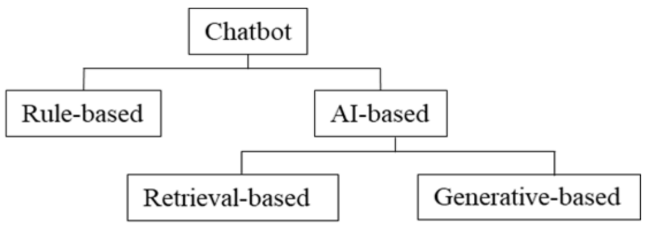
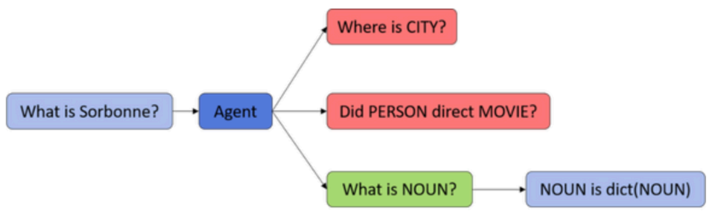
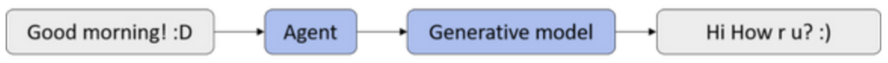


Figure 10: Classification of chatbots based on design techniques

Retrieval-Based Chatbot



Generative Chatbot



3.7 Performance assessment

Another area of challenge in conversational agents is ensuring their ethical use. As AI technology progresses and conversational agents become more sophisticated, there is a risk that they may be used to deceive or manipulate users. This has led to increased discussion around ethical considerations in conversational agents, including issues such as transparency, user privacy, and the potential for harm. In response, researchers and developers have begun to explore ethical guidelines and frameworks for conversational agents to ensure their responsible use. Some examples of these efforts include the IEEE Global Initiative for Ethical Considerations in AI and Autonomous Systems and the Asilomar AI Principles. These initiatives aim to promote ethical use of conversational agents and other AI technologies, while also ensuring their continued development and progress.

4. AI Chatbots in Human Resource Management:

The aim of this chapter is to provide a comprehensive overview of the problem at hand, the goals that we aim to achieve through the implementation of the bot, the constraints that need to be considered during the design process, and the identified user problems that the bot will be addressing. By understanding these factors, we can design a bot that is tailored to the specific needs of the organization, and that can provide employees with the support they need to carry out their work effectively.

4.1 Problem description

One of the main challenges associated with implementing a bot that handles employee queries is ensuring the accuracy of its responses. This requires the bot to have a deep understanding of the organization's policies, procedures, and systems, as well as the ability to accurately interpret and respond to employee queries.

Achieving accuracy requires the bot to be trained on a diverse and representative dataset, which can include historical tickets, chat logs, and other relevant data sources. The bot must also be able to learn from feedback and continuously improve its accuracy over time.

Another key challenge is delivering a positive user experience for employees interacting with the bot. This requires the bot to be easy to use and navigate, with clear and concise instructions and responses. The bot must also be able to adapt to different communication styles and languages, and provide personalized responses based on the employee's specific query.

Integration with existing systems and processes is also important for the successful implementation of a bot that handles employee queries. This requires the bot to be able to access and integrate with relevant data sources, such as HR systems, payroll databases, and other internal systems. The bot must also be able to follow established workflows and procedures, and be able to escalate tickets to the appropriate support agents or departments when necessary.

In order to address these challenges, organizations may need to invest in advanced natural language processing and machine learning technologies, as well as provide extensive training and support to employees and support agents. Regular testing and performance monitoring can also help ensure that the bot is providing accurate and effective responses, and identify areas for improvement.

4.2 Goals

The goal of a bot for query redirection in an employee web page oriented platform is to efficiently handle and redirect employee queries to the appropriate department or support agent. By automating this process, the bot can provide faster and more accurate responses to employee queries, which can help improve employee satisfaction and productivity.

The bot's primary goal is to analyze employee queries using natural language processing and machine learning techniques to understand their intent and provide relevant information or redirect the query to the appropriate department or support agent. The bot should also be able to maintain an accurate and up-to-date knowledge base, which can help improve its accuracy and effectiveness over time.

Another goal of the bot is to provide a seamless and user-friendly experience for employees interacting with the platform. This requires the bot to be easy to use and navigate, with clear and concise instructions and responses. The bot should also be able to adapt to different communication styles and languages, and provide personalized responses based on the employee's specific query.

Ultimately, the goal of a bot for query redirection in an employee web page oriented platform is to provide fast and efficient support for employees, while also improving the overall efficiency and productivity of the organization.

Ideally, the software should :

- Accurate Responses: the bot should be able to provide accurate responses to employee queries and have a high degree of accuracy in understanding the intent behind the queries.

- The bot should also be able to extract relevant information from the employee queries, such as specific dates, names, or numbers, in order to provide more tailored and helpful responses. This requires the bot to have advanced information extraction capabilities, which can be achieved through the use of machine learning and other natural language processing techniques.

- Personalization: The bot should be able to personalize responses based on the employee's specific query and provide relevant information or redirect the query to the appropriate department or support agent.

- Easy to Use: The bot should have an intuitive and user-friendly interface, with clear instructions and responses that are easy to understand and follow.

- Multilingual Support: The bot should be able to support multiple languages and be able to handle queries in different languages.

- Integration with Existing Systems: The bot should be able to integrate with existing systems and processes to provide seamless support and avoid duplication of effort.
- Continuous Learning: The bot should be able to continuously learn and improve its accuracy and effectiveness over time, based on user feedback and interactions.

- Availability: The bot should be available 24/7 to provide support to employees, which can help reduce wait times and improve employee satisfaction.

4.3 Constraints

potential constraints that a bot in an employee web page oriented platform may have include:

- Security and Privacy: The bot should be designed to comply with the organization's security and privacy policies, ensuring that employee data is protected and not shared with unauthorized parties.

- Scalability: The bot should be able to handle a large volume of employee queries and be scalable enough to accommodate future growth in the organization.

- Integration with existing systems and processes: The bot must be able to integrate with existing systems and processes within the organization, such as HR databases or customer relationship management (CRM) systems.

- Language support: The bot should be designed to support multiple languages to cater to the diverse needs of employees.

- Limited conversational capabilities: The bot may have difficulty engaging in complex or nuanced conversations with employees, which could limit its ability to fully address their queries.

- Training data availability: The accuracy of the bot depends on the quality of training data used to develop it. Availability of training data may constrain the bot's performance.

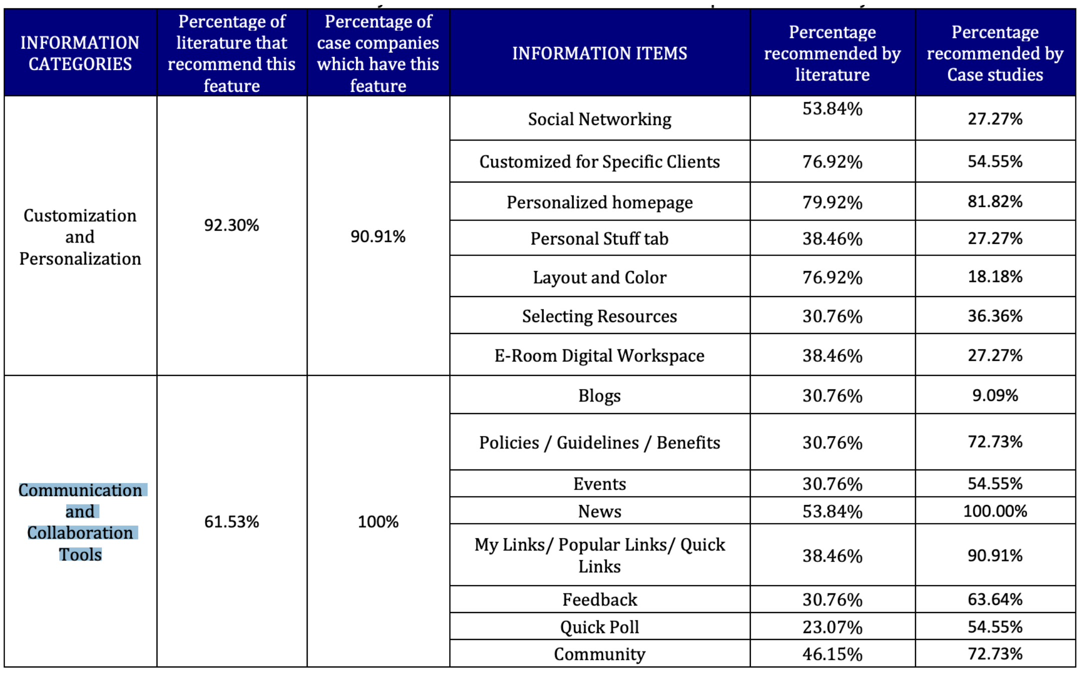
- Budget: The development and implementation of a bot can require significant financial resources, which may be a constraint for some organizations.

4.4 Identified issues

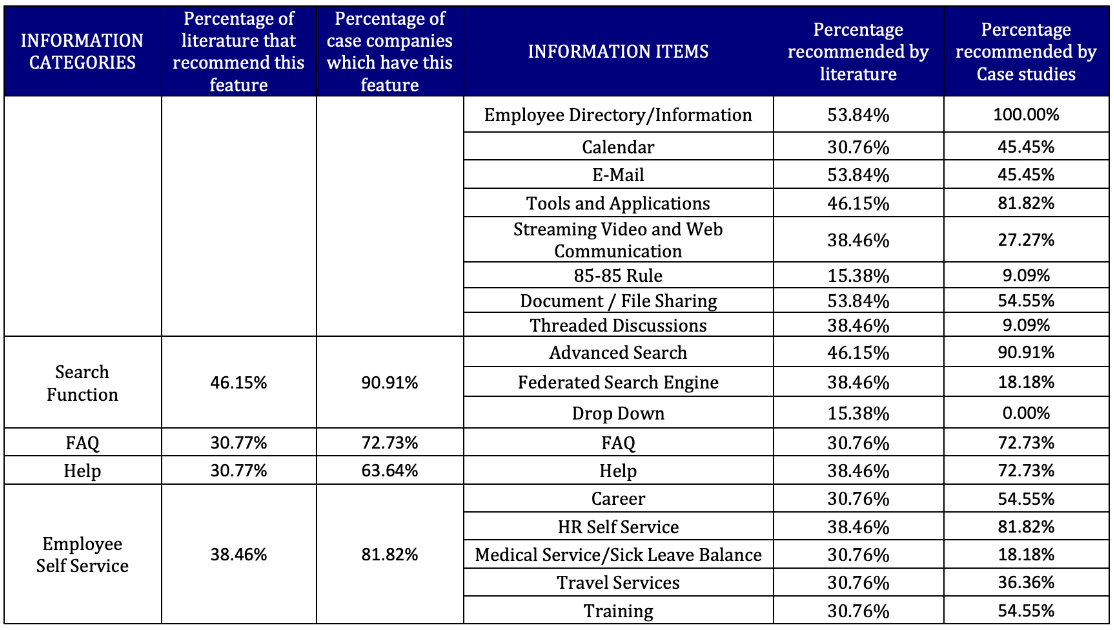
The initial inquiry is to determine "Which user problems should the chatbot be able to handle?” This is a significant challenge since the issues must be common enough to provide ample training data and straightforward enough to prevent potential errors by the chatbot.

1. common enough to provide enough training samples.

2. simple to resolve so as to limit the chatbot’s potential mistakes



INFORMATION CATEGORIES	Percentage of literature that recommend this feature	Percentage of case companies which have this feature	INFORMATION ITEMS	Percentage recommended by literature	Percentage recommended by Case studies
Customization and Personalization	92.30%	90.91%	Social Networking	53.84%	27.27%
			Customized for Specific Clients	76.92%	54.55%
			Personalized homepage	79.92%	81.82%
			Personal Stuff tab	38.46%	27.27%
			Layout and Color	76.92%	18.18%
			Selecting Resources	30.76%	36.36%
			E-Room Digital Workspace	38.46%	27.27%
Communication and Collaboration Tools	61.53%	100%	Blogs	30.76%	9.09%
			Policies / Guidelines / Benefits	30.76%	72.73%
			Events	30.76%	54.55%
			News	53.84%	100.00%
			My Links/ Popular Links/ Quick Links	38.46%	90.91%
			Feedback	30.76%	63.64%
			Quick Poll	23.07%	54.55%
			Community	46.15%	72.73%



INFORMATION CATEGORIES	Percentage of literature that recommend this feature	Percentage of case companies which have this feature	INFORMATION ITEMS	Percentage recommended by literature	Percentage recommended by Case studies
			Employee Directory/Information	53.84%	100.00%
			Calendar	30.76%	45.45%
			E-Mail	53.84%	45.45%
			Tools and Applications	46.15%	81.82%
			Streaming Video and Web Communication	38.46%	27.27%
			85-85 Rule	15.38%	9.09%
			Document / File Sharing	53.84%	54.55%
			Threaded Discussions	38.46%	9.09%
Search Function	46.15%	90.91%	Advanced Search	46.15%	90.91%
			Federated Search Engine	38.46%	18.18%
			Drop Down	15.38%	0.00%
FAQ	30.77%	72.73%	FAQ	30.76%	72.73%
Help	30.77%	63.64%	Help	38.46%	72.73%
Employee Self Service	38.46%	81.82%	Career	30.76%	54.55%
			HR Self Service	38.46%	81.82%
			Medical Service/Sick Leave Balance	30.76%	18.18%
			Travel Services	30.76%	36.36%
			Training	30.76%	54.55%

Figure 11 : Industry Standards and Best Practices – Comparison and Analysis

While communication and collaboration tools in employee portals can be very beneficial, they can also lead to some user problems. Here are some potential issues that can arise:

User problem	Description
Login problems	Users may have difficulty logging into the portal or accessing specific tools within the portal.
Compatibility issues	Certain tools may not be compatible with certain devices or operating systems, leading to technical issues or limited functionality.
Data storage limitations	Some tools may have limitations on the amount of data that can be stored or shared, which can lead to technical issues if users exceed those limitations.
Benefits enrollment	Employees may have difficulty understanding their benefits options, or may have trouble enrolling in benefits programs.
Time off and vacation	Employees may have questions about their vacation time, sick leave, or other time off policies and procedures.
Payroll and compensation	Employees may have questions about their pay, taxes, or other compensation-related issues
Work schedules	Employees may have questions about their work schedules or may need to request time off

Table 1 : List of considered user problems

When employees submit queries to a chatbot designed to provide support, the chatbot may use macros to respond appropriately and attach tags to the query. These tags can help the chatbot quickly identify the nature of the problem and respond effectively.

It is assumed that employees will only report one issue at a time in a single query to the chatbot. This ensures that the chatbot can efficiently address each issue and provide accurate and helpful responses. If employees report multiple issues in a single query, it may be more difficult for the chatbot to identify and address each issue individually, which could lead to longer response times and less effective solutions.

5. Building chatbots

The architecture of the chatbot system is illustrated in Figure 12, which shows that the system retrieves information from its knowledge base and connects it to the inference engine. This connection establishes the relationships throughout the entire system. The data flow design of the architecture provides a clear understanding of the chatbot's concept.

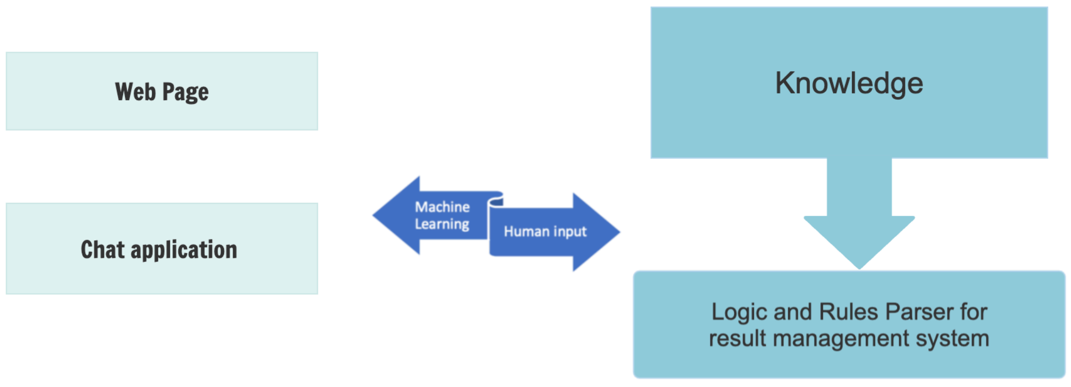


Figure 12 : Anatomy of the chatbot system

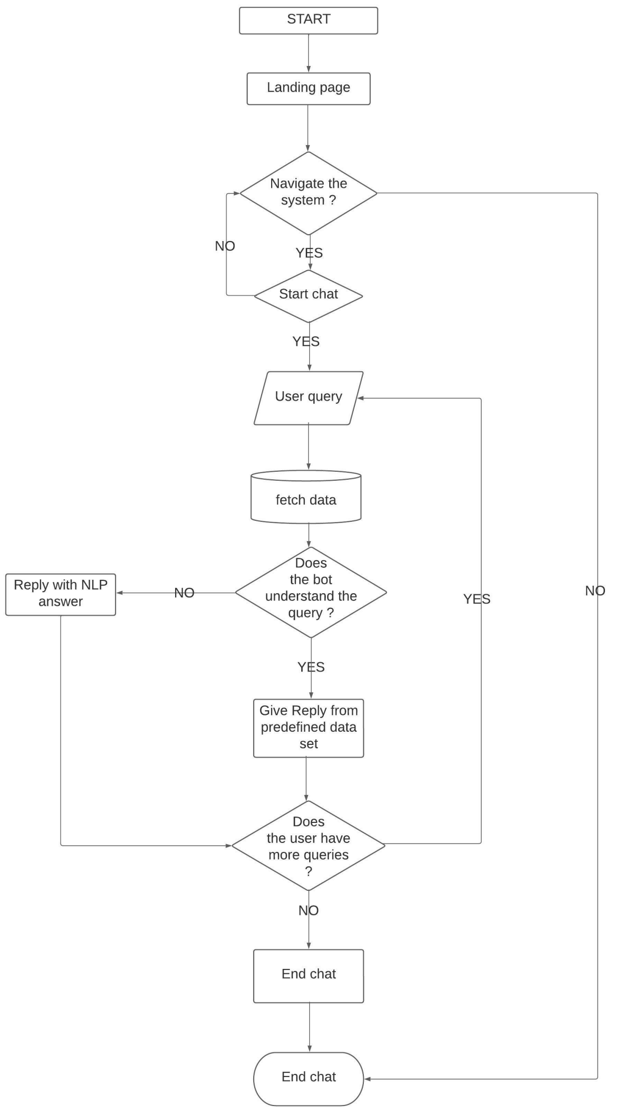


Figure 13 : Proposed system flowchart

5.1 Flask :

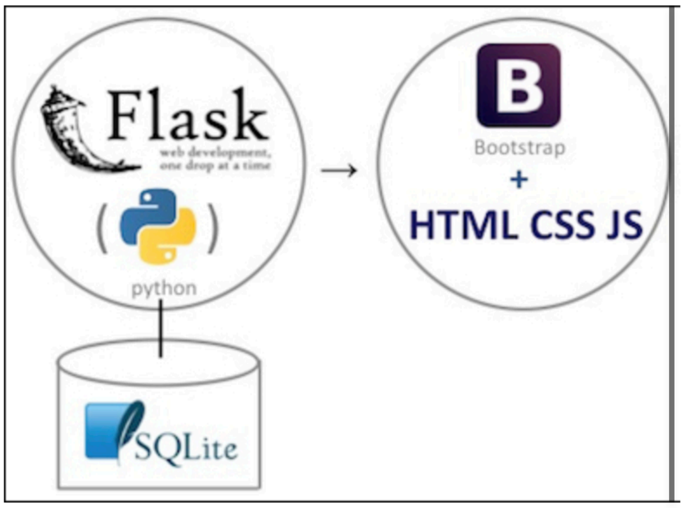


Figure 14 : Flask + Bootstrap + SQLite

Flask is a popular, lightweight, and extensible web application framework written in Python. It provides a simple and efficient way to develop web applications by offering a range of essential tools and features out-of-the-box. Flask is built on the Werkzeug WSGI (Web Server Gateway Interface) toolkit and the Jinja2 templating engine, which together allow developers to create dynamic, scalable, and customizable web applications with ease.

One of the main advantages of Flask is its simplicity, as it does not enforce a specific project structure or require a steep learning curve. This makes it an ideal choice for small to medium-sized projects, as well as for developers who are new to web application development. With its minimalist and modular design, Flask enables developers to build applications by adding only the necessary components, avoiding the complexities and overhead of larger frameworks.

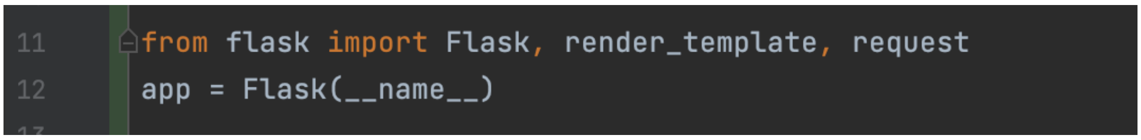
Flask's flexibility is another key strength, as it allows developers to choose from a wide variety of third-party extensions and libraries to enhance the functionality of their applications. This adaptability enables the creation of web applications tailored to specific requirements and ensures that the framework can grow alongside the project.

In the context of developing the chatbot for query redirection in an employee web page, Flask's simplicity and flexibility make it a suitable choice. Its easy-to-use routing system, seamless integration with the Jinja2 templating engine, and compatibility with various databases and machine learning libraries enable the efficient development of a chatbot that can handle user inputs, process natural language queries, and interact with an employee database to provide relevant information.

5.1.1 Flask initialization:

The Flask app initialization is a crucial step in setting up the foundation of the chatbot application. The 'app' variable is created by instantiating the Flask class, which serves as the core component of the Flask application. This instance of the Flask class represents the chatbot application and provides essential functionality for defining routes, handling HTTP requests, and managing application configurations.

The 'app' variable is initialized with the following line of code:



```
from flask import Flask, render_template, request
app = Flask(__name__)
```

Figure 15: app variable initialization

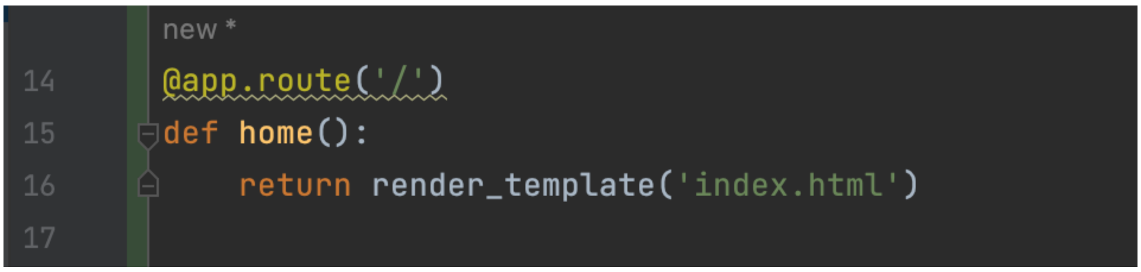
Here, the '**name**' argument passed to the Flask class is used to determine the root path of the application. This is essential for Flask to locate other resources, such as templates and static files, which are used in the chatbot application.

Once the 'app' variable is created, it is used throughout the code to define the application's routes and handle HTTP requests. Routes are the URLs at which users can access different parts of the web application. In the context of the chatbot, the routes are responsible for rendering the user interface and processing user messages. By using the 'app' variable along with route decorators (e.g., @app.route('/')), you can associate specific functions with particular URLs, allowing the chatbot to respond to user requests accordingly.

5.1.2 Route for Home Page:

In Flask, decorators are used to associate specific functions with particular routes or URLs. In this case, the '@app.route('/')' decorator associates the subsequent 'home()' function with the root URL of the application, which is the home page.

Here's the code snippet for the home route:



```
new *
@app.route('/')
def home():
    return render_template('index.html')
```

Figure 16: home route code snippet

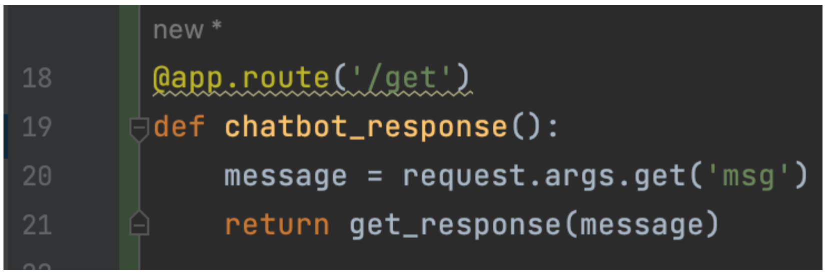
The associated 'home()' function serves as the entry point for users to interact with the chatbot. When users visit the root URL of the application, the 'home()' function is executed, and it returns the rendered 'index.html' template. The 'render_template()' function is a built-in Flask function that takes an HTML template file as an argument and dynamically generates an HTML page to be displayed to the user.

The 'index.html' template serves as the user interface for the chatbot application. It contains input elements for users to enter their messages and interact with the chatbot, as well as a display area to show the chatbot's responses. This user interface allows for seamless communication between the user and the chatbot, enabling the chatbot to receive user inputs, process them, and return relevant responses based on the user's queries.

5.1.3 Route for Chatbot Response:

We use the decorator '@app.route('/get')' this decorator associates the subsequent 'chatbot_response()' function with the '/get' URL of the application, which is specifically designed to handle user messages and generate appropriate responses from the chatbot.

Here's the code snippet for the chatbot response route:



```
new *
@app.route('/get')
def chatbot_response():
    message = request.args.get('msg')
    return get_response(message)
```

Figure 17: chatbot response route code snippet

The associated 'chatbot_response()' function receives user messages as HTTP GET requests. The user's message is passed as a parameter named 'msg' in the request. The 'request.args.get()' function retrieves the value of the 'msg' parameter from the HTTP request, and the resulting message is stored in the 'message' variable.

Once the user's message is obtained, the 'chatbot_response()' function calls the 'get_response()' function, passing the 'message' variable as an argument. The 'get_response()' function is responsible for processing the user's message, interpreting its content, and generating an appropriate response based on the chatbot's understanding of the message. This function utilizes the pre-trained Neural Network model, natural language processing techniques, and database interactions to produce a relevant response for the user.

Finally, the 'chatbot_response()' function returns the generated response as an HTTP response, which is then displayed to the user in the chatbot's user interface. This process allows for a smooth and interactive conversation between the user and the chatbot.

5.1.4 Processing User Messages:

The function used is called ‘get_response()’ this function is responsible for processing user messages, interacting with the employees' database, and generating appropriate responses using the pre-trained Neural Network model.

```python
def get_response(message):
    mydb = pymysql.connect(
        host="localhost",
        user="root",
        password="",
        database="employees",
    )

    mycursor = mydb.cursor()

    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    with open('intents.json', 'r') as json_data:
        intents = json.load(json_data)

    FILE = "data.pth"
    data = torch.load(FILE)

    input_size = data["input_size"]
    hidden_size = data["hidden_size"]
    output_size = data["output_size"]
    all_words = data['all_words']
    tags = data['tags']
    model_state = data["model_state"]

    model = NeuralNet(input_size, hidden_size, output_size).to(device)
    model.load_state_dict(model_state)
    model.eval()

    bot_name = "TFG"
```

In this block of code, the get_response() function initializes various components required to process the user's message and generate an appropriate response. Here's an explanation of each part of the code:

Database connection - Device configuration - Loading intents - Loading model data - Model initialization - Botname

```
def get_employee_info(emp_no):
    mycursor.execute(f"SELECT first_name FROM employees.employees WHERE emp_no = {emp_no}")
    result = mycursor.fetchone()
    if result:
        return result[0]
    else:
        return None

sentence = message
sentence = tokenize(sentence)
X = bag_of_words(sentence, all_words)
X = X.reshape(1, X.shape[0])
X = torch.from_numpy(X).to(device)

output = model(X)
_, predicted = torch.max(output, dim=1)
tag = tags[predicted.item()]

prob = torch.softmax(output, dim=1)[0][predicted.item()]
```

This block of code in the ‘get_response()’ function is responsible for processing the user's message and generating an appropriate response using the pre-trained Neural Network model.

get_employee_info() function: This inner function takes an employee number (emp_no) as an argument, queries the 'employees' database to fetch the first_name of the employee with that number, and returns it. If no employee is found with the given number, it returns None.

Tokenization: The user's message is tokenized using the tokenize() function, which splits the input message into a list of words.

Bag of words: The tokenized message is converted into a bag-of-words representation using the bag_of_words() function. This representation is a fixed-size vector that captures the presence (or frequency) of words in the message.

Model prediction: The bag-of-words representation is fed into the Neural Network model to predict the most suitable response tag. The torch.max() function is used to find the index of the highest value in the output tensor, which corresponds to the predicted tag.

```
prob = torch.softmax(output, dim=1)[0][predicted.item()]
if prob.item() > 0.5:
    for intent in intents['intents']:
        if tag == intent["tag"]:
            if tag == "employee_info":
                pattern = re.compile(r'employee\s+(\d+)', re.IGNORECASE)
                match = pattern.search(" ".join(sentence))
                if match:
                    emp_no = match.group(1)
                    first_name = get_employee_info(emp_no)
                    if first_name is not None:
                        response = intent['responses'][0].format(emp_no=emp_no, first_name=first_name)
                    else:
                        response = "I couldn't find that employee's information."
                else:
                    response = "Please provide an employee number."
            else:
                response = random.choice(intent['responses'])
            return f"{bot_name}: {response}"
else:
    return f"{bot_name}: I do not understand..."
```

This block of code in the ‘get_response()’ function is responsible for generating the chatbot's response based on the predicted tag and its probability.

Checking prediction confidence: The code checks if the probability of the predicted tag is greater than a threshold (0.5 in this case) to ensure that the model is confident enough in its prediction before generating a response.

Iterating through intents: If the confidence threshold is met, the code iterates through the intents defined in the intents.json file to find the intent that matches the predicted tag.

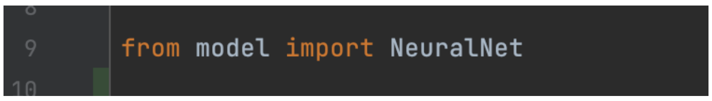
Handling 'employee_info' tag: If the predicted tag is 'employee_info', the code looks for an employee number in the user's message using a regular expression pattern. If an employee number is found, the get_employee_info() function is called to retrieve the employee's first name. The appropriate response is then formatted using the employee's number and first name.

Returning the chatbot's response**:** The chatbot's response is returned as a formatted string that includes the bot's name.

Handling low confidence predictions: If the confidence threshold is not met, the chatbot returns a default response indicating that it does not understand the user's message.

5.2 Neural Network model :

The Neural Network model used in this chatbot application is responsible for processing and understanding user messages. It takes the tokenized and preprocessed messages as input and generates predictions for the appropriate response based on the message content.



```
from model import NeuralNet
```

Figure 18: importing NeuralNet

Key components:

Input Size (input_size): This is the size of the input vector, which represents the bag of words for a given user message. The input size is equal to the length of the 'all_words' list, which contains all unique words in the training dataset. The bag of words is a binary vector where each element corresponds to the presence (1) or absence (0) of a word from 'all_words' in the user message.

Hidden Size (hidden_size): This is the size of the hidden layers in the neural network. The hidden layers are responsible for learning complex patterns and relationships between the input data (user messages) and the output data (predicted response tags). A larger hidden size allows the model to learn more complex patterns, but it can also increase the risk of overfitting and the computational cost.

Output Size (output_size): This is the size of the output vector, which represents the predicted tag probabilities for a given user message. The output size is equal to the number of unique tags in the 'intents.json' file. Each element in the output vector corresponds to the probability of a specific tag being the appropriate response for the input message

Loading the Trained Model: The 'data.pth' file stores the trained model's parameters and related information, such as input_size, hidden_size, output_size, all_words, and tags. The file is loaded using the torch.load() function, and the model's state dictionary is updated with the saved parameters using model.load_state_dict(). This allows the chatbot to utilize the pre-trained model to generate predictions for user messages.

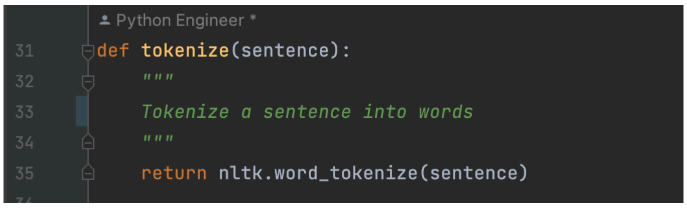
```python
import torch
```

Figure 19: importing torch

5.3 Natural Language Processing (NLP) :

Natural Language Processing (NLP) is a critical component of the chatbot application, as it enables the model to understand and process user messages. The nltk_utils module provides essential NLP functions that facilitate these tasks. The two main functions used from this module are:

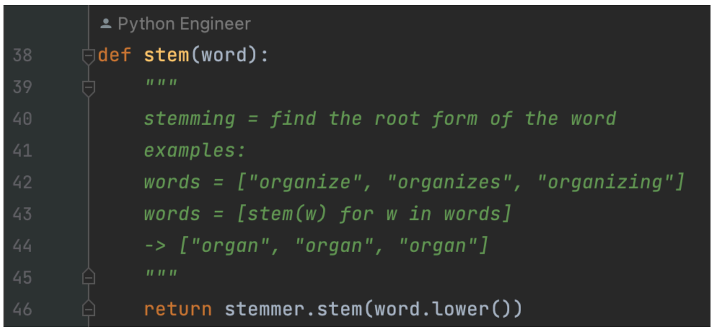
tokenize(): This function is used to break down the user's message into individual words or tokens. Tokenization is a vital preprocessing step in NLP, as it converts raw text into a structured format that can be analyzed and processed by the model. By splitting the message into tokens, the model can recognize and understand individual words, making it easier to determine the meaning and context of the input



```python
def tokenize(sentence):
    """
    Tokenize a sentence into words
    """
    return nltk.word_tokenize(sentence)
```

Figure 20: tokenize function

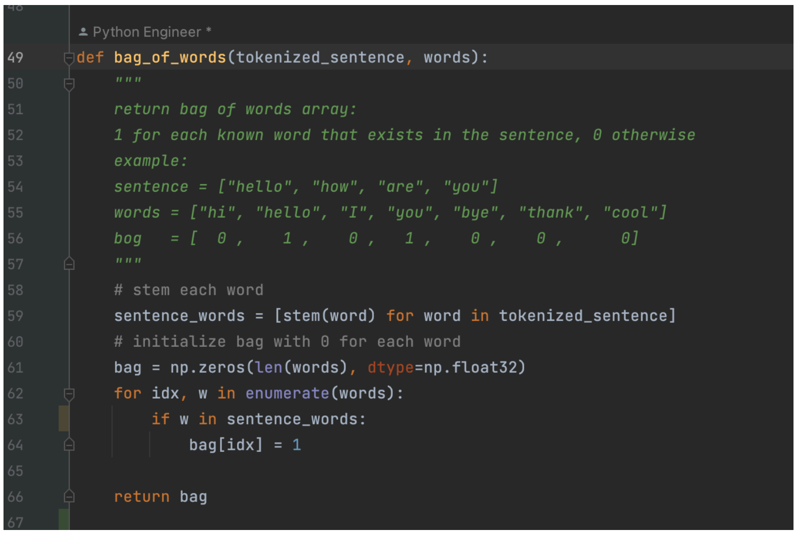
stem(): This function is a utility function for stemming words. Stemming is an NLP technique that aims to reduce a word to its root or base form by removing inflections and derivational affixes. This process helps normalize the text data and reduces the vocabulary size, making it easier for the model to understand and process text input.



```
Python Engineer
def stem(word):
    """
    stemming = find the root form of the word
    examples:
    words = ["organize", "organizes", "organizing"]
    words = [stem(w) for w in words]
    -> ["organ", "organ", "organ"]
    """
    return stemmer.stem(word.lower())
```

Figure 21: stem function

bag_of_words(): This function is used to create a "bag of words" representation of the tokenized message. A bag of words is a simplified representation of the text, where each token is represented as a binary value (1 or 0) indicating its presence or absence in the message. This representation disregards word order and grammar but captures the essential information about which words are present in the input.



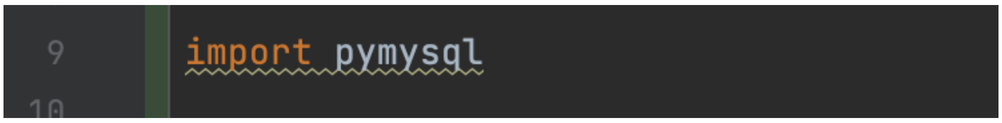
```
Python Engineer *
def bag_of_words(tokenized_sentence, words):
    """
    return bag of words array:
    1 for each known word that exists in the sentence, 0 otherwise
    example:
    sentence = ["hello", "how", "are", "you"]
    words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]
    bog   = [  0 ,    1 ,    0 ,   1 ,    0 ,    0 ,      0]
    """
    # stem each word
    sentence_words = [stem(word) for word in tokenized_sentence]
    # initialize bag with 0 for each word
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1

    return bag
```

Figure 22: bag_of_words function

5.4 Database Integration :

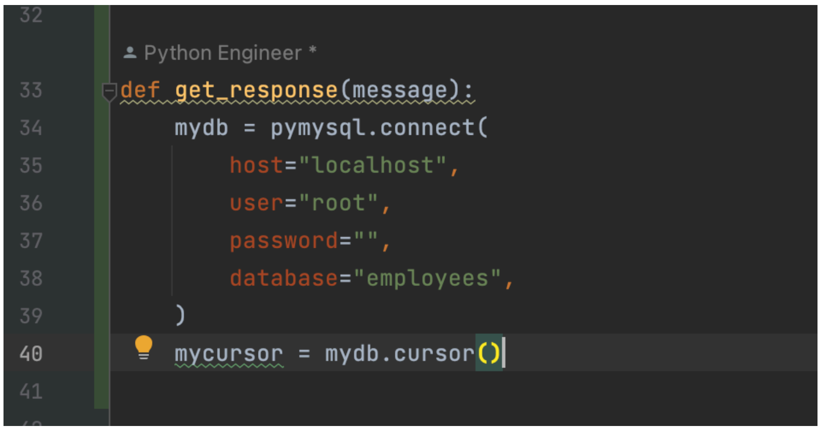
Database Integration is an essential aspect of the chatbot application, as it enables the retrieval of specific employee information when needed. The chatbot is integrated with the employees' database using the pymysql library, a Python MySQL client library that allows for easy interaction with MySQL databases.



```python
import pymysql
```

Figure 23: importing pymysql

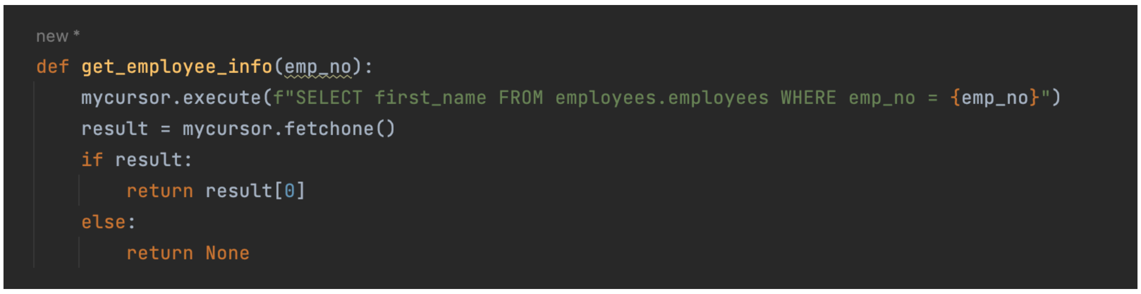
The get_employee_info() function is a crucial component of this integration, as it queries the database for employee information based on the employee number provided by the user. The function performs the following steps:



```python
def get_response(message):
    mydb = pymysql.connect(
        host="localhost",
        user="root",
        password="",
        database="employees",
    )
    mycursor = mydb.cursor()
```

Figure 24: get_response function

Connection to mysql database, the database used is a sample database from : https://dev.mysql.com/doc/employee/en/employees-installation.html.



```
new *
def get_employee_info(emp_no):
    mycursor.execute(f"SELECT first_name FROM employees.employees WHERE emp_no = {emp_no}")
    result = mycursor.fetchone()
    if result:
        return result[0]
    else:
        return None
```

Figure 25: get_employee_info function

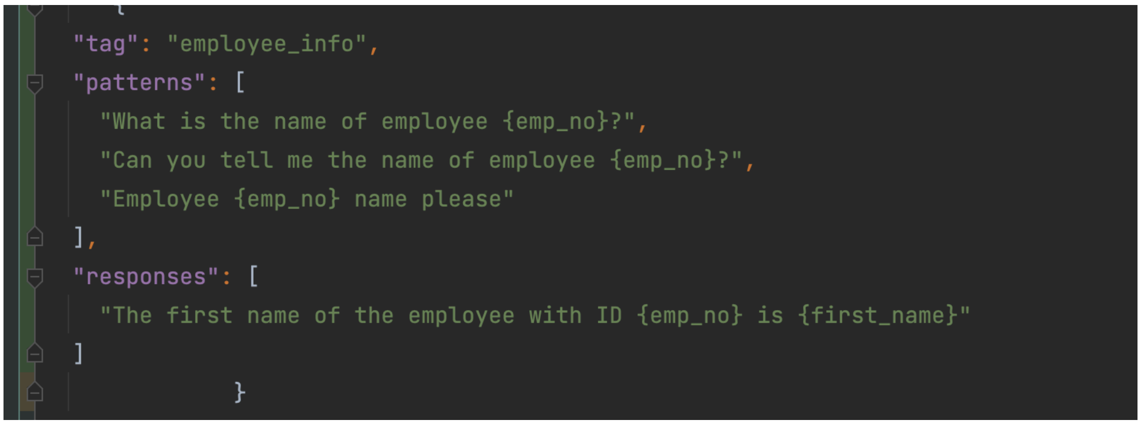
While this function uses a basic SELECT query to fetch the first_name of an employee based on the provided employee number (emp_no), it serves as an example to demonstrate the integration of the chatbot with the employees' database.

In the actual implementation of the chatbot, more complex and diverse queries were used to fetch various types of employee information, such as last names, job titles, department names, and hire dates.

5.5 Query Processing and Response Generation :

When a user sends a message, it is first tokenized and preprocessed using the functions from the nltk_utils module. The tokenization involves breaking the message into individual words, while the bag of words representation converts the tokenized message into a fixed-size input vector for the Neural Network model. This preprocessing is essential to ensure that the model can effectively understand and process the user's message.

The preprocessed input vector is then passed through the trained Neural Network model, which predicts the most appropriate tag for the user's message. The model architecture consists of input_size, hidden_size, and output_size layers. The input_size corresponds to the size of the bag of words representation, while the output_size is the number of unique tags in the 'intents.json' file. The model returns a probability distribution over the possible tags, and the tag with the highest probability is selected as the predicted tag



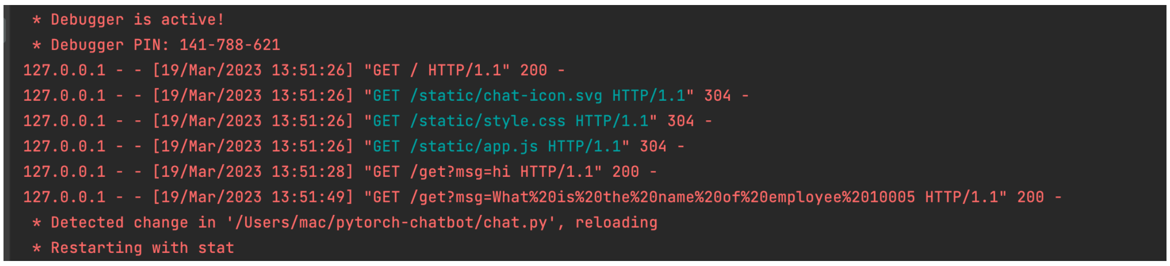
```
    {
    "tag": "employee_info",
    "patterns": [
      "What is the name of employee {emp_no}?",
      "Can you tell me the name of employee {emp_no}?",
      "Employee {emp_no} name please"
    ],
    "responses": [
      "The first name of the employee with ID {emp_no} is {first_name}"
    ]
            }
```

Figure 26: Example of a tag in intents.json

Once the predicted tag is obtained, the system searches through the 'intents.json' file to find the corresponding intent. The 'intents.json' file contains predefined intents, each with a tag, a list of patterns (sample user messages), and a list of responses. Based on the predicted tag, the system selects a random response from the list of available responses associated with the intent.

If the predicted tag requires querying the employees' database (e.g., "employee_info"), the chatbot extracts the necessary information from the user's message (e.g., employee number) and uses the get_employee_info() function to fetch the requested data from the database. The response is then generated by incorporating the retrieved data.

Finally, the generated response is returned as an HTTP response and displayed to the user in the 'index.html' template, enabling seamless interaction between the user and the chatbot.

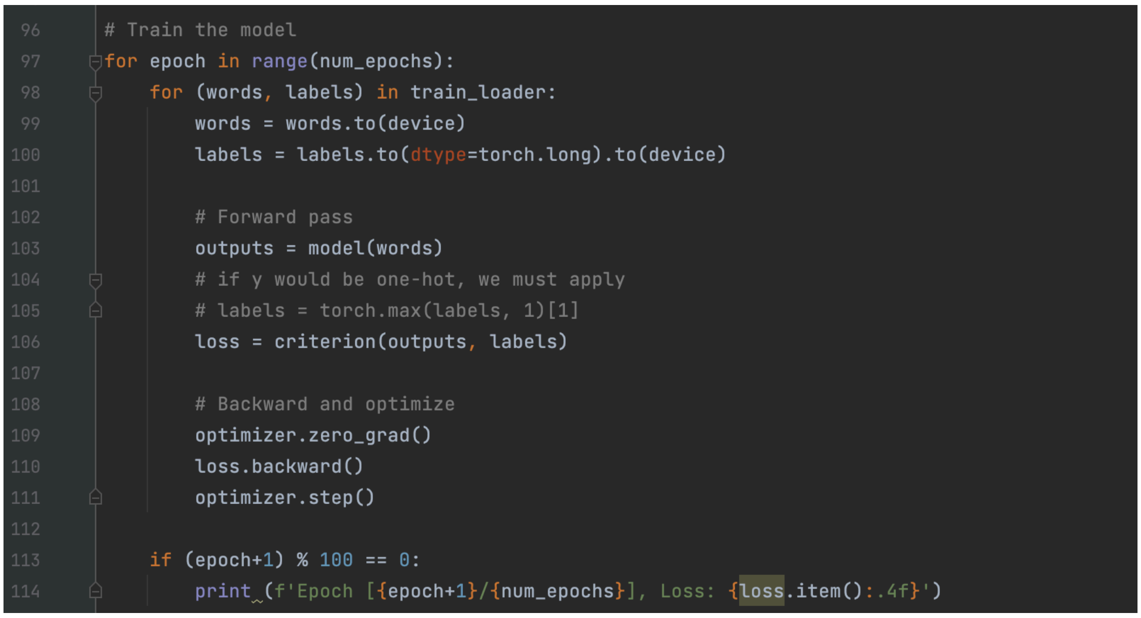


```
 * Debugger is active!
 * Debugger PIN: 141-788-621
127.0.0.1 - - [19/Mar/2023 13:51:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Mar/2023 13:51:26] "GET /static/chat-icon.svg HTTP/1.1" 304 -
127.0.0.1 - - [19/Mar/2023 13:51:26] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [19/Mar/2023 13:51:26] "GET /static/app.js HTTP/1.1" 304 -
127.0.0.1 - - [19/Mar/2023 13:51:28] "GET /get?msg=hi HTTP/1.1" 200 -
127.0.0.1 - - [19/Mar/2023 13:51:49] "GET /get?msg=What%20is%20the%20name%20of%20employee%2010005 HTTP/1.1" 200 -
 * Detected change in '/Users/mac/pytorch-chatbot/chat.py', reloading
 * Restarting with stat
```

Figure 27: example of generated HTTP responses

5.6 Training :

The script performs several key tasks, such as data preparation, model training, and saving the trained model.



```python
# Train the model
for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(dtype=torch.long).to(device)

        # Forward pass
        outputs = model(words)
        # if y would be one-hot, we must apply
        # labels = torch.max(labels, 1)[1]
        loss = criterion(outputs, labels)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    if (epoch+1) % 100 == 0:
        print (f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
```

Figure 28 : training loop for the Neural Network model

The script begins by loading the 'intents.json' file, which contains the intents, patterns, and responses. It tokenizes and stems the patterns to create a list of unique words (all_words) and a list of unique tags (tags). The data is then converted into input-output pairs, where the input is a bag of words representation of the patterns, and the output is the corresponding class label (tag index).

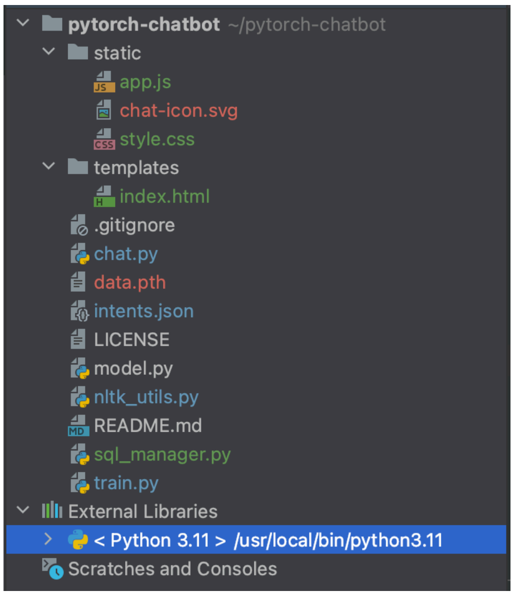
Then the script defines several hyperparameters, such as the number of epochs, batch size, and learning rate for training the Neural Network model. It also sets the input_size, hidden_size, and output_size for the model architecture. The ChatDataset class is created as a subclass of the PyTorch Dataset class to handle the training data. The DataLoader is used to create mini-batches and shuffle the data during training.

The Neural Network model is instantiated and set to run on either GPU (CUDA) or CPU, depending on the availability. The CrossEntropyLoss criterion is used as the loss function, and the Adam optimizer is chosen for updating the model's parameters. The training loop iterates over the specified number of epochs, and for each epoch, it processes the data in mini-batches. The model's forward pass computes the output, which is then compared with the ground truth labels using the loss function. The optimizer performs a backward pass to update the model parameters based on the computed gradients.

After the training process is complete, the final loss is printed to show the model's performance. The model's state_dict (containing the model's parameters), input_size, hidden_size, output_size, all_words, and tags are saved in a dictionary called 'data'. This data is then saved to a file named 'data.pth' using the torch.save() function, which can later be loaded to use the trained model for processing user messages.

6. Deploying the chatbot

In this final chapter, we will demonstrate the deployment of the chatbot and showcase its functionality by interacting with it through a web interface. This will allow users to ask the chatbot various queries and see the chatbot's responses in real-time. The purpose of this chapter is to present the chatbot's practical application and its ability to understand and answer user questions effectively.



```
pytorch-chatbot ~/pytorch-chatbot
  static
    app.js
    chat-icon.svg
    style.css
  templates
    index.html
  .gitignore
  chat.py
  data.pth
  intents.json
  LICENSE
  model.py
  nltk_utils.py
  README.md
  sql_manager.py
  train.py
External Libraries
  < Python 3.11 > /usr/local/bin/python3.11
Scratches and Consoles
```

Figure 29 : Project directory

To create an appealing user interface for our chatbot, we have designed an HTML template that includes essential elements such as a text input field for entering queries, a display area for showing the chatbot's responses, and a send button for submitting user input. The template incorporates CSS styling to ensure the interface is visually appealing and easy to navigate.

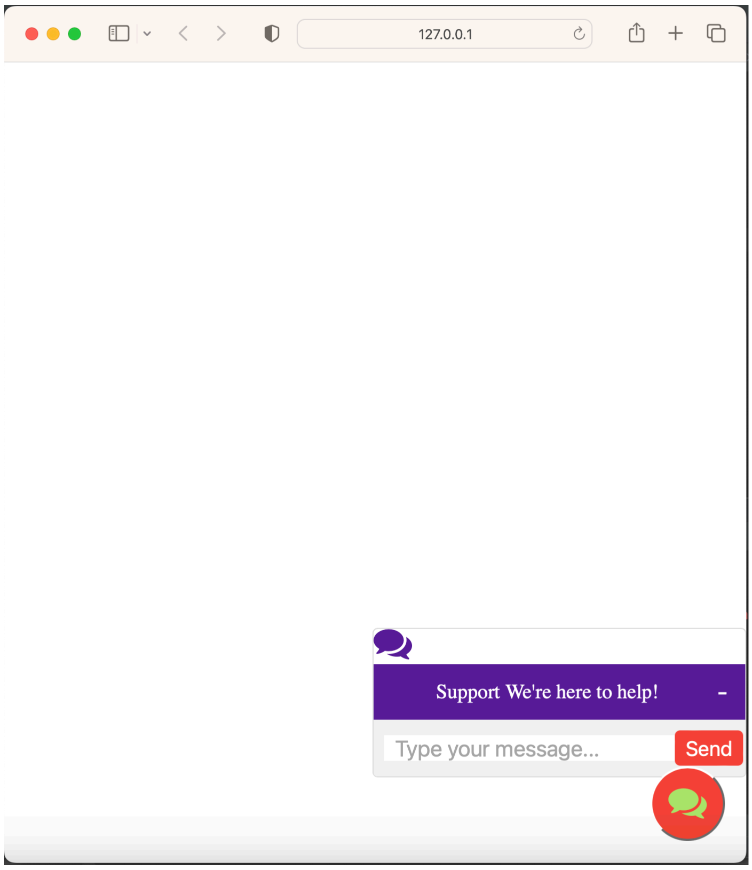
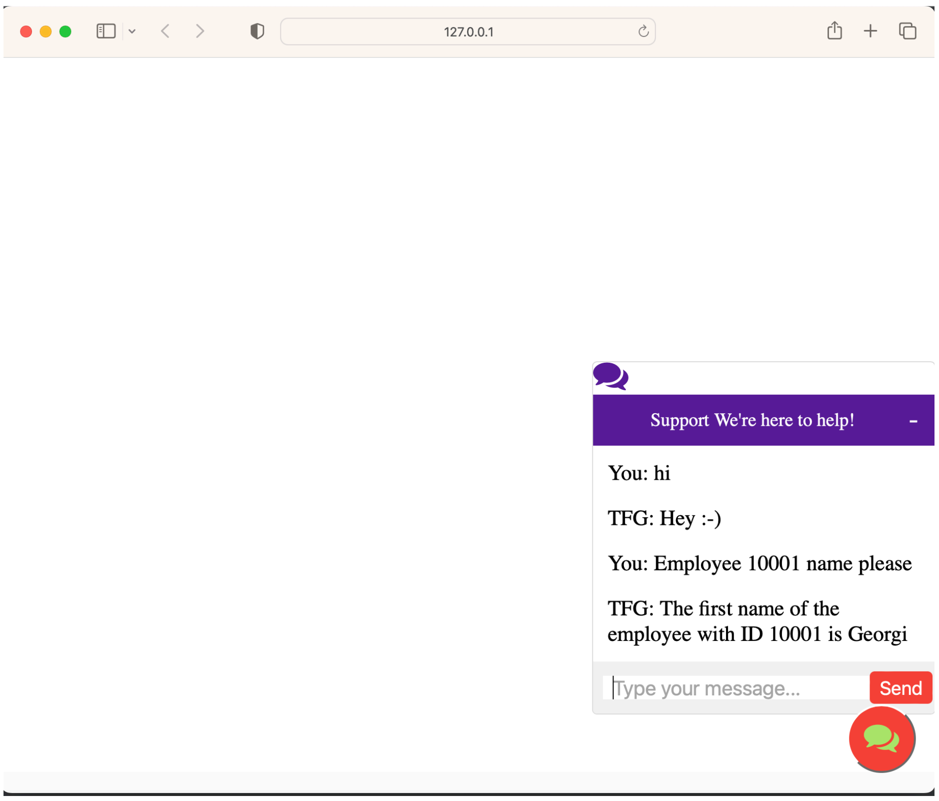


Figure 30 : Chatbot Web Interface.

In addition to the HTML template, we have implemented JavaScript functions to handle user input and manage the interaction between the user and the chatbot. These functions are responsible for capturing user input, sending it to the Flask server via AJAX requests, and updating the display area with the chatbot's responses. This ensures a smooth, real-time interaction between the user and the chatbot without requiring page refreshes.

6.1 Query Processing and Response Generation :

In this section, you can present multiple user interaction examples with the chatbot, showcasing its capabilities and versatility. Here's a sample structure for presenting these interactions:



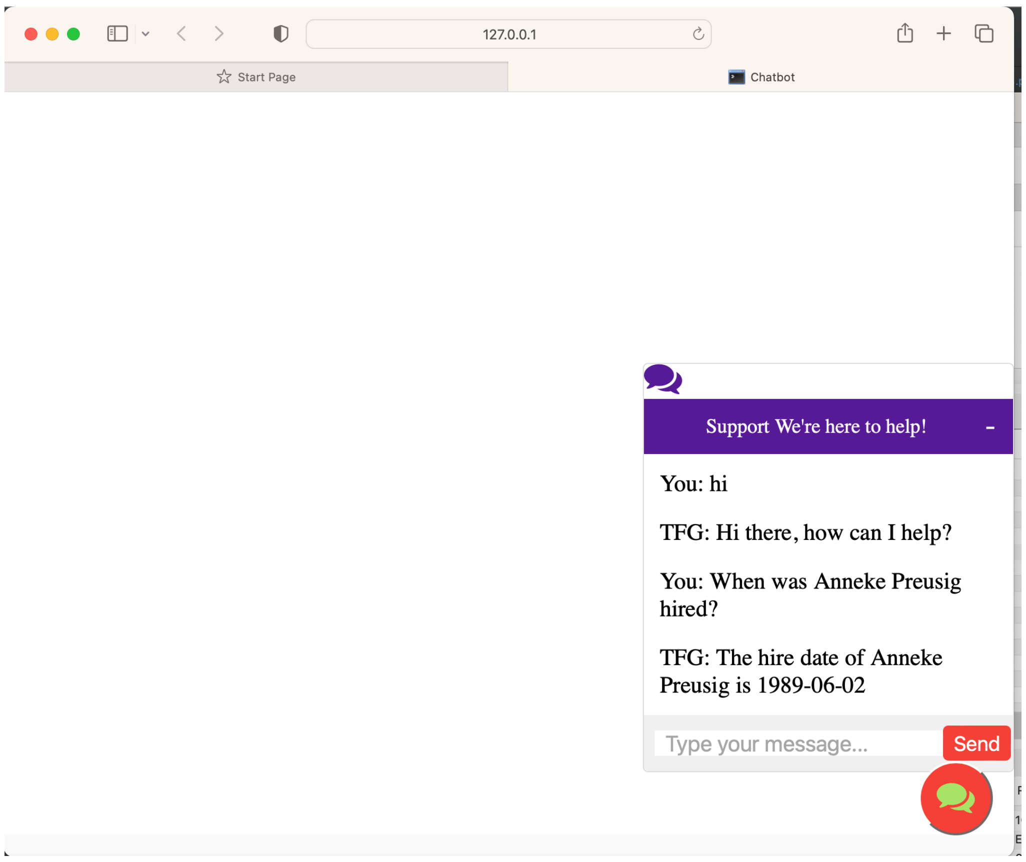
You: hi

TFG: Hey :-)

You: Employee 10001 name please

TFG: The first name of the employee with ID 10001 is Georgi

Figure 31 : Requesting information about the name of an employee with a specific employee number 10001.



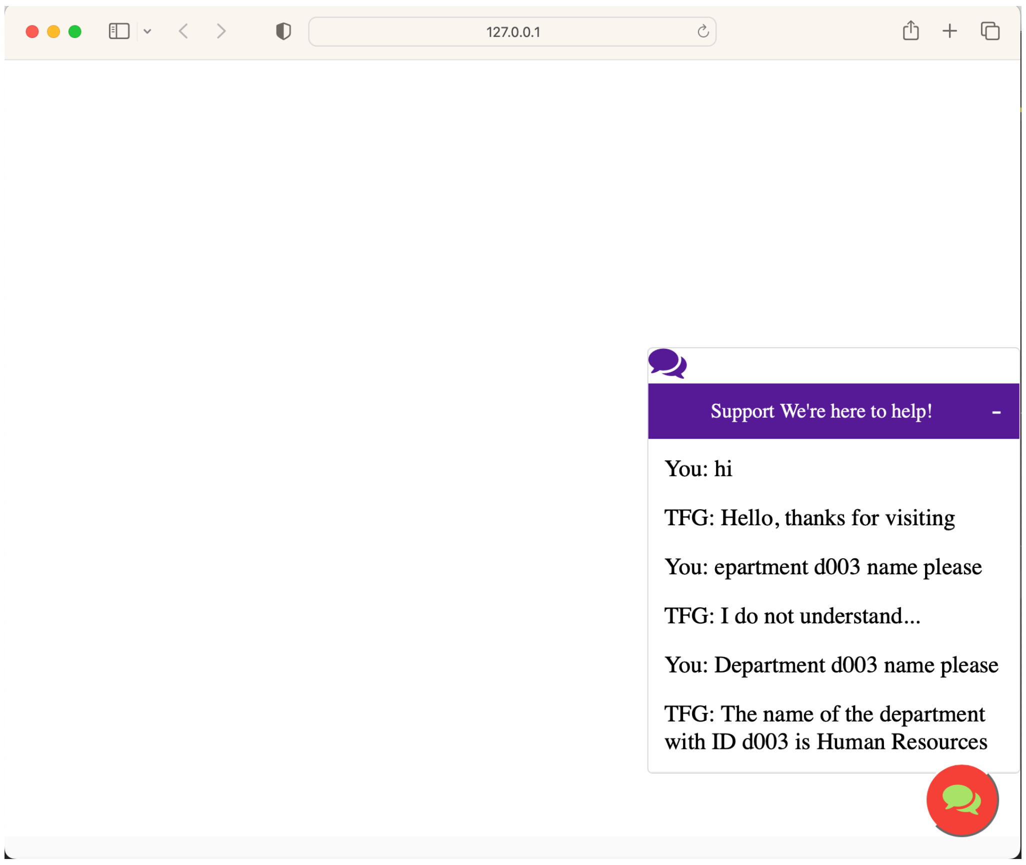
You: hi

TFG: Hi there, how can I help?

You: When was Anneke Preusig hired?

TFG: The hire date of Anneke Preusig is 1989-06-02

Figure 32: Seeking information about the date when a specific person was hired.



You: hi

TFG: Hello, thanks for visiting

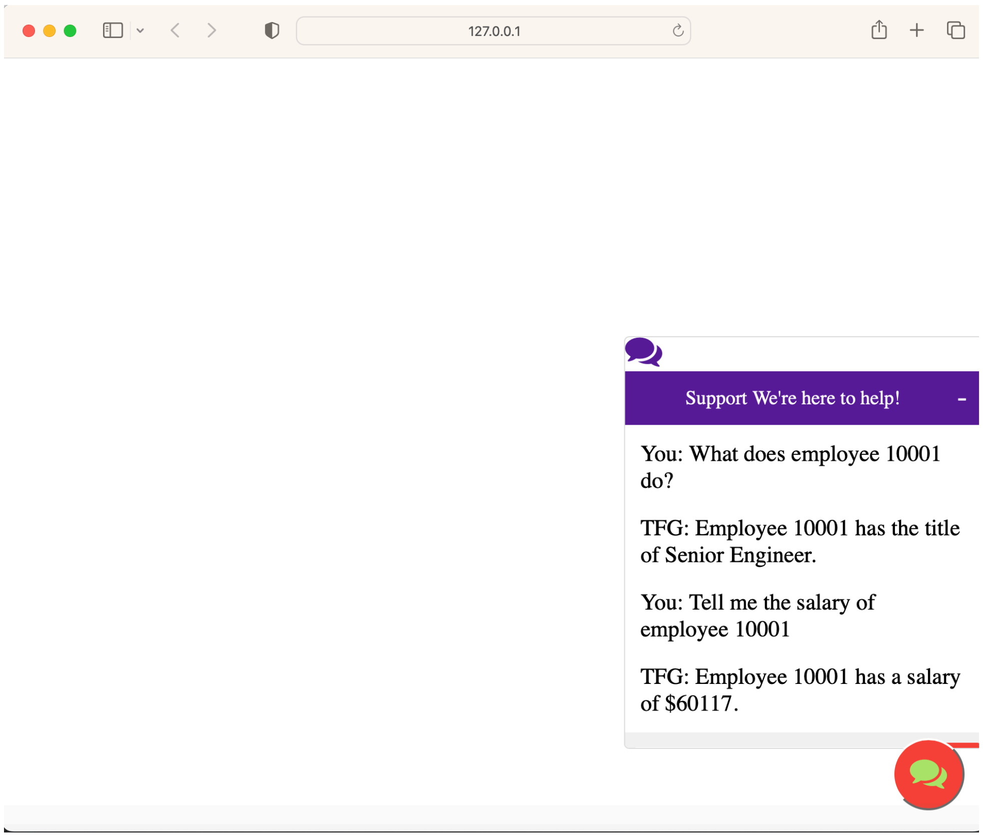
You: epartment d003 name please

TFG: I do not understand...

You: Department d003 name please

TFG: The name of the department with ID d003 is Human Resources

Figure 33: Retrieving the name of a department based on its department number.



You: What does employee 10001 do?

TFG: Employee 10001 has the title of Senior Engineer.

You: Tell me the salary of employee 10001

TFG: Employee 10001 has a salary of $60117.

Figure 34: Retrieving the title of an employee and his salary.

7. Conclusions and future lines of research :

The primary objectives of this project were to develop a chatbot capable of handling employee queries and effectively retrieving relevant information from a MySQL database. The results of this project demonstrate that the developed chatbot has been successful in achieving these objectives.

The chatbot's implementation, which involved the integration of PyMySQL, a deep learning model, and natural language processing techniques, enabled it to effectively communicate with the database and provide accurate responses to a wide range of employee queries.

It increases efficiency by automating the process of handling employee queries, which saves time and effort for human resources and administrative staff, allowing them to focus on other critical tasks. This efficiency may lead to cost savings for the organization.

Furthermore, the chatbot's quick and accurate responses contribute to improved employee satisfaction, which can result in higher retention rates and increased employee engagement. By leveraging the chatbot's automated capabilities, organizations can also reduce human error when responding to employee queries, leading to more accurate and consistent information being provided to employees.

The chatbot is highly scalable, capable of handling a large volume of queries without requiring additional resources. This makes it an ideal solution for growing organizations or those with frequent fluctuations in employee numbers. Additionally, the chatbot offers an accessible method for obtaining information, as employees can interact with it through a familiar chat interface. This convenience may lead to higher adoption rates and increased utilization of the chatbot across the organization.

7.1 Limitations :

Despite the numerous benefits provided by the chatbot, there are some limitations to be acknowledged. One potential limitation is the chatbot's performance in handling complex or ambiguous queries, which may require human intervention for accurate responses. The chatbot relies on predefined intents and patterns, and it might not be able to handle queries outside its training scope, or those that involve subjective interpretations.

Another limitation is the dependency on the quality and completeness of the data used in the project. If the database contains outdated or inaccurate information, the chatbot may provide incorrect responses to employee queries. Moreover, the chatbot's performance may be influenced by the quality of the Natural Language Processing (NLP) model and the training data used to develop it. Insufficient or biased training data may lead to reduced accuracy in the chatbot's responses.

Additionally, the chatbot may face challenges in understanding certain linguistic nuances, such as slang, idioms, or regional dialects, which could affect the accuracy of its responses. The chatbot's performance may also be influenced by the technology or platform used for deployment, as different platforms may have varying levels of compatibility with the chatbot's features and requirements.

Lastly, privacy and security concerns should be considered when implementing the chatbot in an organization. Ensuring that the chatbot handles sensitive employee information securely and in compliance with data protection regulations is essential to maintain employee trust and prevent potential data breaches.

7.2 Future lines of research :

Expanding the scope of the chatbot to handle more complex queries and tasks, making it a more versatile tool for employees.

Incorporating additional data sources, such as external APIs, to provide more comprehensive information and making the chatbot a centralized information hub for employees.

Improving the natural language understanding capabilities of the chatbot by implementing more advanced NLP techniques or fine-tuning the model with domain-specific data, which would lead to better query comprehension and more accurate responses.

Enhancing the user experience by developing a more interactive and user-friendly interface, which could encourage greater adoption and use of the chatbot by employees.

Evaluating the chatbot's performance and impact on employees' productivity and satisfaction through user testing and feedback, allowing for iterative improvements and ensuring that the chatbot continues to meet the needs of its users.

1. Bibliography

[1] Armstrong's Handbook of Human Resource Management Practice" by Michael Armstrong, 2017.

[3] The Human Capital Hub URL : https://www.thehumancapitalhub.com/articles/functions-of-human-resources-how-the-department-works

[3] Narayanan, K. (1998). Technology acquisition, de-regulation and competitiveness: A study of Indian automobile industry. Research Policy, 27(2), 215–228.

[4] Bos, A. S., Pizzato, M., & Zaro, M. A. (2019e). Investigation of Student Attention: The Use of Virtual Reality in Computing Education. Tear: Journal of Education, Science and Technology, 8, 3. https://periodicos.ifrs.edu.br/index.php/tear/index https://doi.org/10.35819/tear.v8.n2.a3586.

[5] SelectHub. (2021). HRIS Integration: Why Your System Needs to Play Nice with Others. Retrieved from https://www.selecthub.com/hris/hris-integration/

[6] Kavanagh, M. J., & Johnson, R. D. (2018). Human resource information systems: Basics, applications, and future directions. Sage Publications.

[7] ichrm URL : https://icehrm.com/blog/features-of-an-effective-hris/

[8] Noe, R. A., Hollenbeck, J. R., Gerhart, B., & Wright, P. M. (2017). Fundamentals of human resource management. McGraw-Hill Education.

[9] Human Resource Information Systems: Basics, Applications, and Future Directions" by Michael J. Kavanagh, Richard D. Johnson, and Talya Bauer

[10] Kavanagh, M. J., Thite, M., & Johnson, R. D. (2019). Human Resource Information Systems: Basics, Applications, and Future Directions (Third ed.). SAGE Publications, Inc.

[11] The URL to Agency for Healthcare Research and Quality (AHRQ) : https://www.ahrq.gov/

[12] Human Resource Information Systems: Basics, Applications, and Future Directions" by Michael J. Kavanagh, Richard D. Johnson, and Dianne Willis Thite, published by Sage Publications in 2019.

[13] ogrnostic URL : https://orgnostic.com/blog/best-hris/

[14] Tojib, D., Sadiq, S. A., & Reynolds, P. (2006). Employee portals and knowledge management: An empirical investigation. Journal of Knowledge Management, 10(3), 135-149. doi: 10.1108/13673270610670892

[15] Lee, S. M., & Kim, H. J. (2009). User satisfaction with business-to-employee portals: Conceptualization and scale development. Computers in Human Behavior, 25(1), 48-61. doi: 10.1016/j.chb.2008.05.005

[16] Remus, U. (2006). Employee portal projects: Complex, time and cost-consuming, with a high risk of failure.

[17] Brown, T., Bond, J., Balaji, S., Brown, J., & Lusch, R. (2007). Developing business-to-employee (B2E) portals: Exploring the drivers and barriers. Journal of Electronic Commerce in Organizations

[18] Werts, C.E., Linn, R.L., Jöreskog, K.G., 1974. Intraclass reliability estimates: testing structural assumptions. Educational and Psychological Measurement 34. White, C., 2003. Determining Enterprise Portal ROI. in: DM Review.

[19] Damsgaard, J., Scheepers, R., 1999. A stage model of intranet technology implementation and management. In: Proceedings of the 7th European Conference on Information Systems, June 23–25, Copenhagen, Denmark.

[20] “An Empirical Investigation of Employee Portal Success" by Benbya et al. (2004)

[21] Benbya, H., Belbaly, N., & Van Alstyne, M. (2004). An empirical investigation of employee portal success

[22] Weizenbaum, J. (1966). ELIZA – a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36-45. Available at https://web.stanford.edu/class/cs124/p36-weizenabaum.pdf

[23] Abdul-Kader, S. A., & Woods, J. C. (2015). Survey on chatbot design techniques in speech conversation systems. International Journal of Advanced Computer Science and Applications

[24] Vodolazova, T., & Kucherov, D. (2020). The Use of Chatbots in the Educational Process. In Proceedings of the 2nd International Conference on Intelligent Human Systems Integration (IHSI 2020): Integrating People and Intelligent Systems (pp. 326-332).

[25] Kusumadewi, S., & Hidayanto, A. N. (2021). The Use of Decision Tree Algorithm to Develop a Chatbot Application for Pregnant Women. Journal of Functional Morphology and Kinesiology

[26] Rashid, M. A., Hasan, M. M., Hoque, A. S. M. L., & Mahmud, S. M. (2021). BERT-based sentiment analysis: A software engineering perspective. In 2021 IEEE International Conference on Electro/Information Technology (EIT) (pp. 365-369)

[26] URL de BERT : https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-bert/

[27] XLNet: Generalized Autoregressive Pretraining for Language Understanding" by Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le.
[28] Ait Ouakrim, D., Benamar, N., Boumhidi, I., El Moussami, H., & Ait Ouahman, A. (2022). A review on the use of artificial intelligence for the optimization of photovoltaic systems. Electronics,

[29] Devopedia. "Flask." Devopedia, https://devopedia.org/flask.