COMP1921 : Advanced Topics in Data Science and AI

Amaan khan
BSc Computing
001239369-X

Word Count : 3498

Axure RP 10 : Prototype

Google drive files

Contents

Introduction

This essay explores the development of a virtual healthcare assistant, focusing on technical implementation and human-computer interaction (HCI) principles. Part A examines data preparation, model selection, and evaluation, addressing challenges in creating an AI model. Part B highlights the HCI aspects, including user-centred design, accessibility, and iterative feedback integration, to ensure the system is intuitive, inclusive, and effective for non-technical users.

Part A:

Data Preparation and Model Choice

Data

Clean data is critical for building a reliable NLP model, which serves as the backbone for developing a healthcare app. During the app's development, missing numerical values were replaced with means, while categorical entries were filled with modes to maintain demographic consistency. Text fields were standardised by removing punctuation and converting to lowercase, ensuring uniformity and reducing errors caused by case sensitivity in user input. Duplicate rows, which could introduce bias or redundancy, were eliminated to improve the dataset's integrity and app performance. These data-cleaning steps ensured the dataset was well-structured and optimised for accurate model training and app functionality, following recognised best practices for machine learning projects(Han, Kamber, & Pei, 2011).

(see figure 1)

Diverse Data

A diverse and representative dataset is essential for building an inclusive NLP model that performs reliably across real-world scenarios. The dataset includes individuals from different age groups, genders, and socio-economic backgrounds, addressing variations in health literacy and communication styles. By ensuring this diversity, the assistant can cater to a broad user base, providing accurate and unbiased advice. To ensure a balance between common and rare health conditions, data augmentation was used to increase the presence of less common categories. For example, while conditions like colds were naturally frequent, rarer conditions, such as autoimmune disorders, were added proportionally to prevent the model from focussing too much on common conditions. This balance helps the model adapt to both frequent and less common situations.

The dataset also prioritised linguistic diversity, including symptom descriptions that range from simple terms like “fever” to more technical phrases like “chronic fatigue syndrome.” This variety makes the system accessible to users with different levels of health knowledge and communication styles. To reflect real-world trends, seasonal illnesses such as colds and flu were given emphasis, aligning with their natural prevalence. These combined measures created a strong and inclusive dataset, improving the model’s ability to make accurate and fair predictions across a wide range of cases (Feng, et al., 2021).

Ensuring Representativeness

A representative dataset is crucial for building an NLP model that mirrors real-world healthcare situations. A representative dataset ensures it includes common conditions, symptoms, and demographic trends, helping the virtual assistant give accurate advice for various cases. To reflect real-world patterns, the dataset was compared with public health data to match the frequency of different conditions. Common illnesses, like colds and flu, were included more often to reflect their high occurrence, while rarer conditions were added in proportion to their prevalence. This alignment ensures the model’s predictions are both realistic and useful (Little, 2019).

Symptom categories were also balanced to avoid overrepresenting specific types. The frequency of each symptom was reviewed, and underrepresented categories were expanded using synthetic data generation. This ensured the model could effectively manage both common and less frequent conditions, improving its adaptability (Zhang L. L., 2019).

To account for patient diversity, the dataset included descriptions suitable for various levels of health literacy, from simple terms like “fever” to detailed descriptions like “persistent cough and shortness of breath.” This diversity allows the system to understand inputs from users with different ways of expressing symptoms, making it more accessible and user-friendly (Jurafsky, 2023). By addressing these real-world patterns and user diversity, the dataset forms a strong base for a reliable and trustworthy virtual healthcare assistant (Acock, 2018).

Model Choice

BioBERT was chosen for its focus on biomedical language, which is essential for understanding medical terms and handling vague or unstructured patient inputs. Pre-trained on datasets like PubMed, BioBERT accurately distinguishes between terms like “flu” and “cold” and interprets phrases like “feeling unwell.” Fine-tuning was applied to align the model with the project dataset, improving its ability to handle specific symptom descriptions and adapt to new inputs. This ensures the model can address complex patient communication while staying flexible for new scenarios.

Other models were considered but not selected. Clinical BERT is suitable for structured clinical notes but struggles with conversational or unstructured text. GPT-based models are versatile and handle a wide range of tasks but require significant computational resources and lack domain-specific medical expertise, which is crucial for this project. BioBERT’s specialised focus, adaptability, and efficiency made it the best choice. It meets the project’s requirements for processing diverse patient descriptions and providing accurate, context-aware responses (Lee, 2020).

(See Figure 2)

Model Application

After selecting BioBERT for its expertise in processing biomedical language, the model was fine-tuned to classify patient symptom descriptions. The dataset was split into two parts: 85% for training and 15% for testing. This split allowed the model to learn from most of the data while keeping some aside to evaluate how well it worked on new inputs.

Symptom descriptions were processed using the BioBERT tokenizer, which converts text into numerical tokens that the model can read. For example, a short input like “mild fever” was turned into a brief sequence of numbers, while longer inputs like “persistent cough with shortness of breath” were cut down to a fixed length of 128 tokens. Shorter inputs were padded to meet this length, ensuring all data was consistent.

The model’s architecture was modified for classification by adding a layer that aligned with the symptom categories in the dataset. The training was done using the AdamW optimiser with a learning rate of 5e-5 and cross-entropy loss. The model went through the data in batches over three cycles, called epochs, improving its ability to correctly classify symptoms, even when the inputs were unclear or complex.

(see Figures 3 - 5)

Continuous Learning

The system is designed to handle a variety of patient inputs, ranging from detailed medical descriptions to vague or poorly structured statements. Instead of relying solely on static datasets, it learns dynamically from anonymised user interactions, capturing input patterns and analysing feedback on its responses. This continuous feedback loop supports iterative improvement, enabling the model to adapt its predictions and improve its ability to interpret ambiguous or incomplete descriptions effectively (Fiebrink & Gillies, 2018).

Over time, the assistant develops the capability to deliver increasingly tailored responses by learning from real-world interactions. By adapting to individual users' needs, preferences, and communication styles, it provides more personalised and accurate advice, making the system not only functional but also user-centric. This iterative approach ensures the system evolves in a way that remains highly relevant to diverse user bases (Jurafsky & Martin, 2023).

The system also identifies broader patterns, such as seasonal illnesses, and adjusts its recommendations to reflect these trends. For example, when low-confidence predictions arise, it prompts clarifying follow-up questions or advises consulting a healthcare professional. By incorporating these adaptive mechanisms and learning over time, the assistant ensures it delivers meaningful, accurate, and context-aware support to its users.

Model Evaluation and Challenges

The model's performance in providing symptom advice was evaluated using four key metrics: accuracy, precision, recall, and F1 scores, as these offer a detailed understanding of its strengths and weaknesses. Accuracy measures the proportion of correct predictions out of all predictions made, providing an overall view of how well the model performs. Precision focuses on the relevance of the predictions, showing how often the predicted cases are correct, which is essential for minimising false positives. Recall examines how many of the actual relevant cases were successfully identified by the model, making it crucial for understanding whether the system captures all valid instances. The F1 score balances precision and recall into a single value, offering an overall measure of performance, particularly useful for datasets with imbalanced labels.

The results revealed significant variability. The model achieved an accuracy of 24%, indicating that only a quarter of the predictions were correct. Precision for the "common cold" was 0.80, reflecting an over-reliance on this frequent condition, but its recall was just 0.10, showing that many relevant cases were missed. For rare conditions, such as "migraine" or "stomach bug," both precision and recall were below 0.20, and F1 scores consistently reached 0.00, highlighting the model's inability to handle less common scenarios effectively. These metrics demonstrate that while the model performs reasonably well for frequent conditions, it struggles significantly with rarer ones, underlining the need for improvement before deployment in real-world settings.

(see figures 6 - 7)

Tackling vague input

The system's ability to handle vague or incomplete patient descriptions is critical in delivering accurate symptom advice. Patient inputs often lack detail or use non-medical language, making them challenging to interpret. For example, descriptions such as "feeling sick" or "stomach pain" provide insufficient context for precise diagnosis without further clarification.

The model addresses these challenges by incorporating confidence thresholds to assess the reliability of its predictions. Inputs with low confidence trigger follow-up questions designed to gather additional information. For instance, when faced with a vague input like "feeling unwell," the system may ask targeted questions about specific symptoms, such as fever or nausea. This interactive approach reduces ambiguity, improving the accuracy of the advice provided.

Additionally, the model uses contextual embeddings from pre-trained NLP models, such as BioBERT, to interpret variations in language. This allows it to handle non-specific terms, ensuring accessibility for users with varying levels of health literacy.

Despite these strategies, ambiguous inputs remain a significant challenge, as low-confidence predictions can still lead to errors. To mitigate this, the system prioritises asking clarifying questions or deferring to human professionals when ambiguity persists, ensuring that users receive appropriate guidance even in unclear scenarios (He, 2023).

Preventing Misleading Medical Advice

Ensuring the system avoids providing misleading medical advice is a critical requirement for patient safety. The model uses a confidence threshold to assess the certainty of its predictions. If the confidence level falls below the threshold, the system avoids presenting advice as definitive and instead suggests clarifying questions or encourages the user to consult a healthcare professional. Low-confidence predictions, such as for overlapping symptoms like “fever” and “cough,” trigger mechanisms that prioritise safety. For example, if the input aligns with symptoms of multiple conditions, such as “common cold” or “flu,” the system presents both possibilities while advising further investigation. In cases of critical conditions, such as heart attack or stroke symptoms, the system is programmed to flag these inputs and recommend immediate professional intervention. By implementing these safeguards, the system ensures that it does not rely solely on uncertain predictions, reducing the risk of harmful advice (Tighe, 2024).

(see figure 8)

Insights from Confusion Matrix Analysis

The confusion matrix revealed important details about the model’s performance, highlighting significant issues with incorrect predictions. It showed that the model often classified a wide range of inputs as “common cold,” regardless of the actual symptoms. While 80% of “common cold” cases were correctly identified, rarer conditions like “migraine” or “stomach bug” were classified correctly less than 5% of the time. This clear bias towards frequent conditions limits the model’s ability to provide accurate advice for less common cases, reducing its overall reliability.

The analysis also showed the model struggled to distinguish between conditions with similar symptoms. For instance, inputs such as “fever” and “fatigue” were often misclassified as “common cold” instead of “flu” or other conditions. These findings demonstrate that the model’s focus on frequent conditions harms its ability to handle a variety of cases. This over-reliance on common labels results in poor performance for rarer conditions, making the system less useful in real-world healthcare settings, where accurate and precise differentiation is crucial (Tighe, 2024).

(see figure 9)

Addressing Dataset Imbalance

Dataset imbalance was a significant factor in the model’s poor performance, particularly for rare conditions. Labels such as “common cold” appeared far more frequently in the training data, resulting in over-prediction of these conditions. For example, the label “common cold” accounted for over 40% of predictions, while rarer conditions like “migraine” or “stomach bug” had prediction rates below 5%. This imbalance skewed the model’s learning process, leading to poor recall for less common cases.

The overrepresentation of frequent labels caused the model to favour these outputs, even when symptoms indicated other conditions. Misclassification patterns from the confusion matrix confirmed that rare labels were often ignored, impacting the system’s ability to provide accurate advice. This imbalance highlights the importance of using techniques like oversampling rare conditions or synthetic data generation to ensure fair representation during training. By addressing the imbalance, the model can improve its performance across all condition categories (Salmi, 2024).

Part B

Designing for Non-Technical Users

Intuitive Design and Simplified Navigation for Accessibility

The app is designed to be intuitive and accessible, particularly for patients with limited technological experience. Its minimalist interface eliminates unnecessary distractions, focusing solely on essential features to enhance navigation and comprehension. To guide users in describing their symptoms, the app provides a single input field accompanied by a clear example, such as "e.g., chills," offering clarity and reducing uncertainty. Additionally, for users who may find typing difficult or unfamiliar, a dropdown menu with prefilled symptom options is included, ensuring users can easily select from recognisable options rather than relying on memory. These features align with usability principles that advocate reducing visual clutter, lowering cognitive load, and prioritising "recognition over recall," making the app both efficient and user-friendly (Nielsen, 1994)

(see figures 10-11)

The app's navigation is designed for simplicity and accessibility, featuring two large green buttons—one for moving forward and another for going back. Their size and clarity ensure ease of use for patients unfamiliar with touchscreens or those with limited dexterity. To further enhance usability, the app integrates universally recognisable icons, providing visual cues for key functions. These elements improve accessibility by helping users with low literacy or non-native speakers quickly understand the app’s features without relying solely on text. This design approach emphasises clear, clickable buttons and intuitive navigation, ensuring users can proceed or correct actions without confusion (Horton, 2010).

(see figure 12)

Visual Symptom Mapping and Humanised User Interaction

The app incorporates a body diagram feature that allows patients to tap on specific body areas to indicate where they are experiencing symptoms. This visual tool is particularly effective for patients who may struggle to describe symptoms in words or lack familiarity with medical terms. For example, a patient experiencing pain in their head can simply tap that region, bypassing the need for complex explanations. This design choice aligns with research on the effectiveness of visual representations in improving user comprehension and interaction. The body diagram also adheres to WCAG's (2018) guidelines by providing a universally accessible method of communication, ensuring inclusivity for patients of varying literacy levels or language proficiencies (Ware, 2012), (WCAG, 2018)

(see figure 13)

In addition to the body diagram, the app maintains a warm and conversational tone throughout, such as using phrases like, "Let’s figure this out together," to make the experience feel more personal and less intimidating. This approach follows Norman's (2004) principles, which recommend humanising digital interfaces to build trust and encourage engagement. Together, these features—visual aids for clarity and inclusivity, and a patient-centred tone—equip users with minimal technical knowledge to confidently and effectively communicate their symptoms while navigating the system.

Progressive Symptom Description with Guided Input

To design an interface that helps users confidently describe their symptoms, the app employs a step-by-step approach called Progressive Disclosure, which ensures users are not overwhelmed by information. The process begins with a body diagram, where users can tap on the area of their concern, such as the head for headache-related symptoms. This visual feature is particularly helpful for users who may struggle to articulate their symptoms verbally or are unfamiliar with medical terminology. After selecting a body part, the app displays common symptoms associated with that area, such as "Headache," "Dizziness," or "Vision issues." Each symptom is accompanied by simple, everyday explanations, such as "A sensation of pain or discomfort in the head," along with clear icons, making the information accessible to users with limited medical knowledge (Horton, 2010).

The app then guides users through a series of specific questions about their symptoms, such as rating pain severity on a scale from 1 to 10 using colour-coded faces, indicating how long they've experienced the symptoms, and identifying any possible triggers. Users are also given the option to add additional symptoms they may have missed earlier. By breaking the process into smaller, manageable steps, the app reduces cognitive load and fosters user confidence. With clear visuals, simple language, and guiding prompts, the interface ensures that even users with minimal medical or technical knowledge can describe their symptoms accurately and without hesitation (Nielsen, 1994).

(see figures 13-16)

Clear Results Display and Symptom Mapping with Actionable Guidance

The predicted condition, such as "Common Cold," is displayed prominently at the top in bold text to ensure users can immediately identify it. A brief explanation in plain language, like "A viral infection that affects your nose and throat," avoids medical jargon and provides reassurance. The inclusion of a confidence level indicator, such as "High Likelihood" or a percentage (e.g., "85% confident"), adds transparency to the results and builds trust in the system. This approach aligns with usability principles that emphasize the importance of clear communication and user trust in health applications (Zhang Y. L., 2019).

A visual symptom mapping feature adds another layer of clarity by helping users connect their reported symptoms to the diagnosis. This includes a simple diagram that highlights affected areas of the body, such as the throat or nose for a common cold. Visual aids like this improve comprehension, particularly for users with limited health literacy. Practical advice is provided in the "Suggested Action" section, offering clear steps like "Keep Hydrated: Drink water, teas, or soups" and "Stay Rested: Get plenty of sleep." These actionable steps align with design principles that advocate for user-friendly and accessible health information (Garcia-Retamero, 2013).

User Feedback and Iteration

Feedback Collection Methods

Designing a virtual healthcare assistant that effectively meets user needs requires gathering feedback through multiple methods. Surveys, user testing, focus groups, and system analytics work together to provide a clear understanding of user experience and guide continuous improvements.

Surveys

Surveys are an efficient way to collect feedback from a wide range of users. They include both structured questions, like rating the ease of use on a scale, and open-ended prompts, such as asking users to describe any difficulties. For example, users might answer, "How easy was it to describe your symptoms?" and provide detailed suggestions. Surveys help identify common trends and unique insights that can inform system improvements (Bargas-Avila, 2011).

User Testing

User testing involves observing users as they interact with the assistant. This can include tasks like entering symptoms or reviewing results. Using think-aloud protocols, where participants describe their thoughts while using the system, highlights barriers such as unclear instructions or confusing layouts. Recorded sessions provide detailed data that can guide specific design adjustments (Nielsen, 1994).

Focus Groups

Focus groups allow small groups of users to discuss their experiences with the assistant. These discussions often uncover valuable feedback, including suggestions that individual methods might miss. For example, groups segmented by age or language ability can offer unique perspectives on the assistant's usability and tone. Focus groups are particularly useful for exploring how different user demographics interact with the system (Morgan, 1996).

System Analytics

System analytics track how users navigate the assistant, providing objective data to complement user feedback. Metrics like drop-off rates, time spent on specific screens, or frequently used features help identify usability issues. For instance, a high drop-off rate during symptom entry may indicate confusing instructions, prompting targeted revisions. Combining analytics with qualitative feedback ensures a well-rounded approach to improvement (Albert, 2013).

Iterative Design Process

The iterative design process ensures continuous refinement by integrating user feedback into prototype updates. For example, challenges like difficulty typing symptoms may lead to features such as voice-to-text functionality or dropdown menus with predefined options. These updates are tested in usability sessions to validate their effectiveness before implementation (Krug, 2014; Nielsen, 1994).

To meet diverse user needs, tailored features are introduced. Elderly users benefit from larger text sizes, simplified navigation, and voice input, while non-native speakers require multilingual support and universal icons. Tech-savvy users might prefer advanced features like confidence levels or medical references. These adaptations make the assistant more inclusive and user-friendly (Horton & Lynch, 2010; Ware, 2012).

Revised prototypes are tested with both previous and new participants to ensure unbiased feedback. Incremental updates are deployed in controlled environments to confirm practicality and alignment with user needs before a full rollout (Doshi-Velez & Kim, 2017).

Ensuring Inclusivity and Accessibility

Inclusivity is an important part of designing the virtual healthcare assistant. Feedback is collected from people of different ages, backgrounds, ethnicities, and levels of health knowledge. This approach ensures the assistant works for everyone, including groups that are often overlooked. It follows best practices to create a fair and unbiased system (Sarkar, 2024).

Accessibility focuses on making the assistant easy to use for people with different needs. Features include voice input for users who have trouble typing, larger text for those with vision problems, and compatibility with tools like screen readers. The assistant also uses clear icons and supports multiple languages to help non-native speakers and those with low literacy. Its tone is friendly and supportive. For example, instead of saying "Symptom unclear," it might say, "Let's gather more information to better understand your symptoms," which makes users feel reassured and supported (Venkatesh, 2021).

Continuous Learning and Tailored Responses

By analysing user interactions—such as symptom inputs, corrections, and feedback—the system identifies recurring challenges, like difficulties in phrasing symptoms. To address these issues, it introduces tooltips with example inputs and retrains its natural language processing model to better interpret diverse descriptions.

This approach reflects the principles of human-centred machine learning, which emphasises aligning algorithms with human contexts and needs. Human-centred machine learning integrates user feedback into the machine-learning process, ensuring systems remain relevant and usable. By incorporating these principles, the assistant not only improves its accuracy but also enhances user satisfaction by providing more intuitive and tailored responses (Fiebrink, Introduction to the Special Issue on Human-Centered Machine Learning, 2018).

Ultimately, this iterative feedback loop between users and algorithms demonstrates how hybrid human-machine systems can achieve better results than working independently. The assistant becomes increasingly user-friendly, showcasing the power of continuous learning to bridge technical innovation with practical application.

Conclusion

This essay has looked at how to create a virtual healthcare assistant by combining advanced technology with a focus on users' needs. Careful preparation of data, choosing the right model, and improving the system through ongoing learning help make it accurate and adaptable. Ensuring the assistant is inclusive and accessible allows it to support a wide range of users, including those with limited health knowledge or technology skills. By using feedback, simple design principles, and user-focused machine learning, the assistant shows how AI can be used effectively in healthcare to provide safe, reliable, and personalised support.

References

Acock, A. C. (2018). *A Gentle Introduction to Stata*. Retrieved from https://www.stata.com/bookstore/gentle-introduction-to-stata/

Albert, W. &. (2013). *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Morgan Kaufmann.

Bargas-Avila, J. A. (2011). Old wine in new bottles or novel challenges? A critical analysis of empirical studies of user experience. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, (pp. 2689–2698). doi:10.1145/1978942.1979336

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021). A Survey of Data Augmentation Approaches for NLP. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 10.

Fiebrink, R. &. (2018). Introduction to the Special Issue on Human-Centered Machine Learning. *ACM Transactions on Interactive Intelligent Systems*, 1-7. doi:10.1145/3205942

Garcia-Retamero, R. &. (2013). Communicating Health Risks with Visual Aids. *Current Directions in Psychological Science*, 392–399.

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

He, Z. Y.-N. (2023). "Nothing Abnormal": Disambiguating Medical Reports via Contrastive Knowledge Infusion. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, (pp. 1234–1245). doi: 10.18653/v1/2023.acl-main.123

Horton, S. &. (2010). *Web Style Guide: Basic Design Principles for Creating Web Sites*. Yale University Press.

Initiative, W. A. (2018). *Web Content Accessibility Guidelines (WCAG) 2.1*. World Wide Web Consortium (W3C). Retrieved from https://www.w3.org/WAI/standards-guidelines/wcag/

Jurafsky, D. &. (2023). *Speech and Language Processing*. Stanford University. Retrieved from https://web.stanford.edu/~jurafsky/slp3/

Lee, J. Y. (2020). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 1234–1240. doi: 10.1093/bioinformatics/btz682

Little, R. J. (2019). *Statistical Analysis with Missing Data*. Wiley. doi: 10.1002/9781119482260

Morgan, D. L. (1996). Focus Groups. *Annual Review of Sociology*, 129–152. doi:10.1146/annurev.soc.22.1.129

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

Salmi, M. A. (2024). Handling Imbalanced Medical Datasets: Review of a Decade of Research. *Artificial Intelligence Review*, 273. doi: 10.1007/s10462-024-10884-2

Sarkar, S. P. (2024). Inclusive Digital Health Interventions: Bridging Gaps for Equity. *BMC Global Public Health*. doi:10.1186/s44263-024-00050-9

Tighe, P. G. (2024). Artificial Intelligence and Patient Safety: Promise and Challenges. *PSNet Perspectives*. doi:10.1001/psnet.2024.0001

Venkatesh, V. B. (2021). *Designing for Accessibility: The Role of Empathy in Technology Interventions*. doi:: 10.1007/978-3-030-78462-1_6

Ware, C. (2012). *Information Visualization: Perception for Design*. Elsevier. doi: 10.1016/C2010-0-68350-4

Zhang, L. L. (2019). A Review on Deep Learning Applications in Prognostics and Health Management. *IEEE Access*. doi:10.1109/ACCESS.2019.2952848

Zhang, Y. L. (2019). Designing Health Information Technologies for Patients: A Review of the State of the Art. *Health Informatics Journal*, 1381–1398. doi:10.1177/1460458219849223

Appendix A

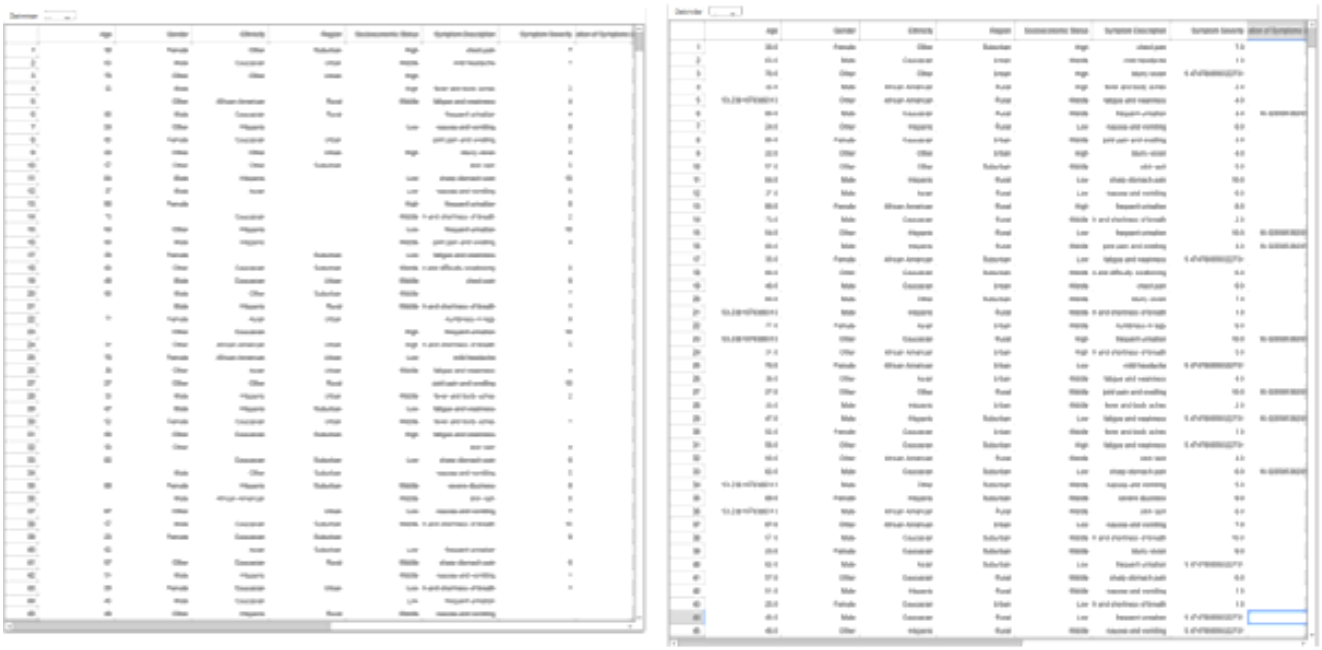

Figure 1: Cleaned Dataset – Before and After

```
# Import necessary libraries
from transformers import BertTokenizer, BertForSequenceClassification
import torch

# Load the fine-tuned model and tokenizer
tokenizer = BertTokenizer.from_pretrained("biobert_tokenizer")
model = BertForSequenceClassification.from_pretrained("biobert_model")

# Set the model to evaluation mode
model.eval()

# Map predicted label to condition and remedy
condition_mapping = {
    0: ("Allergic Reaction", "Take antihistamines and avoid allergens."),
    1: ("Anxiety", "Practice deep breathing or consult a therapist."),
    2: ("Arthritis", "Apply a warm compress and take anti-inflammatory medications."),
    3: ("Asthma", "Use your inhaler and seek fresh air."),
    4: ("COVID-19", "Isolate, rest, and monitor symptoms. Consult a doctor if severe."),
    5: ("Common Cold", "Drink plenty of fluids and rest."),
    6: ("Diabetes", "Monitor your blood sugar levels and follow your prescribed treatment."),
    7: ("Flu", "Rest, stay hydrated, and take fever reducers."),
    8: ("Food Poisoning", "Stay hydrated and avoid solid foods until symptoms improve."),
    9: ("Gastroenteritis", "Drink oral rehydration solutions and rest."),
    10: ("Hypertension", "Reduce salt intake and practice relaxation techniques."),
    11: ("Kidney Stones", "Drink plenty of water and take prescribed pain relief."),
    12: ("Migraine", "Rest in a dark, quiet room and take prescribed medications."),
    13: ("Muscle Strain", "Apply ice, rest, and gently stretch the muscle."),
    14: ("Sinus Infection", "Use steam inhalation and drink warm fluids.")
}

# Function for manual symptom testing
def predict_condition():
    # Ask the user to input their symptoms
    symptom_description = input("Please describe your symptoms: ")

    # Tokenise the input
    inputs = tokenizer(
        symptom_description,
        return_tensors="pt",
        padding="max_length",
        truncation=True,
        max_length=128
    )
```

Figure 2: Code Snippet – Symptom Classification and Remedies

```
4]: # Load BioBERT Tokenizer and Model
    tokenizer = BertTokenizer.from_pretrained("dmis-lab/biobert-base-cased-v1.1")
    model = BertForSequenceClassification.from_pretrained(
        "dmis-lab/biobert-base-cased-v1.1", num_labels=len(np.unique(y))
    )

    # Set device to GPU if available
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.to(device)
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at dmis-lab/biobert-base-cased-v1.1 and are newly initializ
ed: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
4]: BertForSequenceClassification(
      (bert): BertModel(
        (embeddings): BertEmbeddings(
          (word_embeddings): Embedding(28996, 768, padding_idx=0)
          (position_embeddings): Embedding(512, 768)
          (token_type_embeddings): Embedding(2, 768)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (encoder): BertEncoder(
          (layer): ModuleList(
            (0-11): 12 x BertLayer(
              (attention): BertAttention(
                (self): BertSdpaSelfAttention(
                  (query): Linear(in_features=768, out_features=768, bias=True)
                  (key): Linear(in_features=768, out_features=768, bias=True)
                  (value): Linear(in_features=768, out_features=768, bias=True)
```

Figure 3: Tokenization Phase of the Model

```
]: EPOCHS = 3

for epoch in range(EPOCHS):
    model.train()
    total_loss = 0
    print(f"Epoch {epoch + 1}/{EPOCHS}")

    for step, batch in enumerate(train_loader):
        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["labels"].to(device)

        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        total_loss += loss.item()

        loss.backward()
        optimizer.step()

        if step % 10 == 0:
            print(f"  Step {step}/{len(train_loader)} - Loss: {loss.item():.4f}")

    avg_loss = total_loss / len(train_loader)
    print(f"Epoch {epoch + 1} completed. Average Loss: {avg_loss:.4f}")
```

Figure 4: Training Code of the Model

```
Epoch 1/3
  Step 0/80 - Loss: 2.6995
  Step 10/80 - Loss: 2.6679
  Step 20/80 - Loss: 2.4892
  Step 30/80 - Loss: 2.7207
  Step 40/80 - Loss: 2.6024
  Step 50/80 - Loss: 2.5358
  Step 60/80 - Loss: 2.5128
  Step 70/80 - Loss: 2.7297
Epoch 1 completed. Average Loss: 2.6524
Epoch 2/3
  Step 0/80 - Loss: 2.7012
  Step 10/80 - Loss: 2.5362
  Step 20/80 - Loss: 2.6429
  Step 30/80 - Loss: 2.6435
  Step 40/80 - Loss: 2.8186
  Step 50/80 - Loss: 2.4965
  Step 60/80 - Loss: 2.6904
  Step 70/80 - Loss: 2.4594
Epoch 2 completed. Average Loss: 2.6351
Epoch 3/3
  Step 0/80 - Loss: 2.6999
  Step 10/80 - Loss: 2.4674
  Step 20/80 - Loss: 2.6290
  Step 30/80 - Loss: 2.3426
  Step 40/80 - Loss: 2.6089
  Step 50/80 - Loss: 2.6992
  Step 60/80 - Loss: 2.5037
  Step 70/80 - Loss: 2.6219
Epoch 3 completed. Average Loss: 2.6260
```

Figure 5: Training Completion of the Model

```
Accuracy: 0.24

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         9
           1       0.00      0.00      0.00         9
           2       0.00      0.00      0.00        13
           3       0.00      0.00      0.00        13
           4       0.00      0.00      0.00        11
           5       0.24      1.00      0.39        54
           6       0.00      0.00      0.00        17
           7       0.00      0.00      0.00        14
           8       0.00      0.00      0.00         7
           9       0.00      0.00      0.00        13
          10       0.00      0.00      0.00        13
          11       0.00      0.00      0.00        12
          12       0.00      0.00      0.00        10
          13       0.00      0.00      0.00        19
          14       0.00      0.00      0.00        11

    accuracy                           0.24       225
   macro avg       0.02      0.07      0.03       225
weighted avg       0.06      0.24      0.09       225
```

Figure 6: Model Evaluation Summary

	Input	Expected	Predicted	Confidence	Correct
0	pain	Unknown Condition	Common Cold	0.210000	False
1	severe chest pain and sweating	Heart Attack	Common Cold	0.240320	False
2	difficulty breathing	Asthma	Common Cold	0.219702	False
3	headache	Migraine	Common Cold	0.205116	False
4	I feel unwell	Unknown Condition	Common Cold	0.195772	False

Figure 7: Test Cases Results

```
Additional check for emergency conditions
 predicted_label in [15, 16, 17]:  # Emergency conditions
  print("⚠ Warning: Your symptoms indicate a medical emergency. Please seek immediate medical attention.")
```

Figure 8: Implementation of Medical Emergency Warning Code

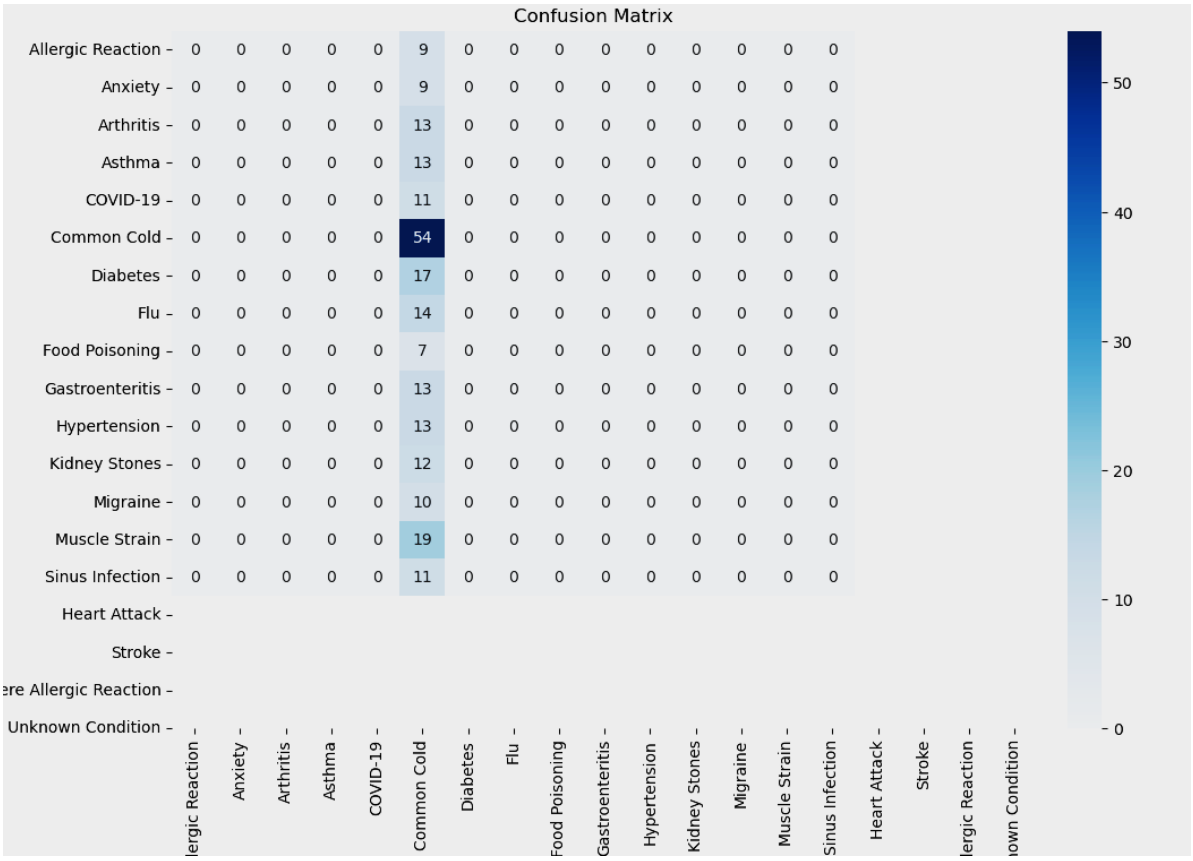

Figure 9: Confusion Matrix Results

Appendix B

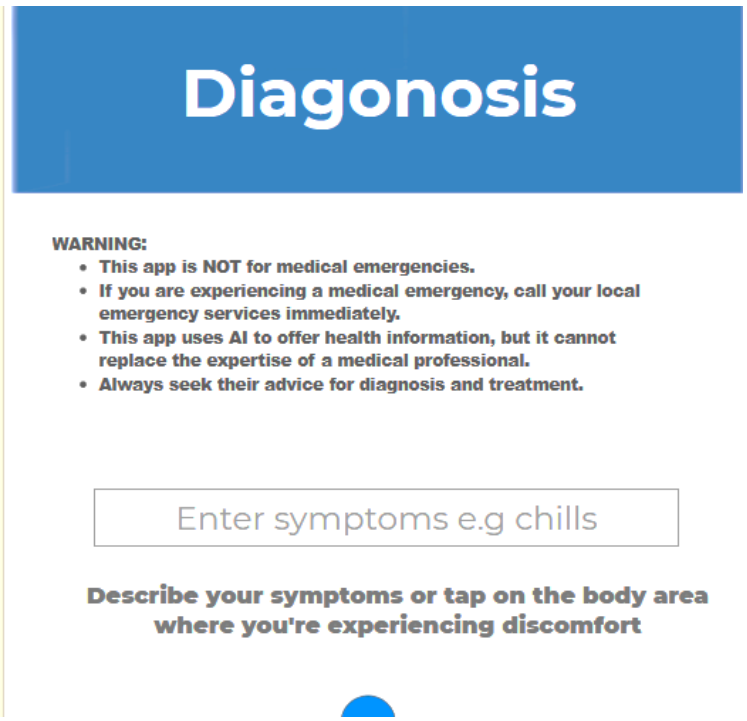

Figure 10: Search Bar with Hints

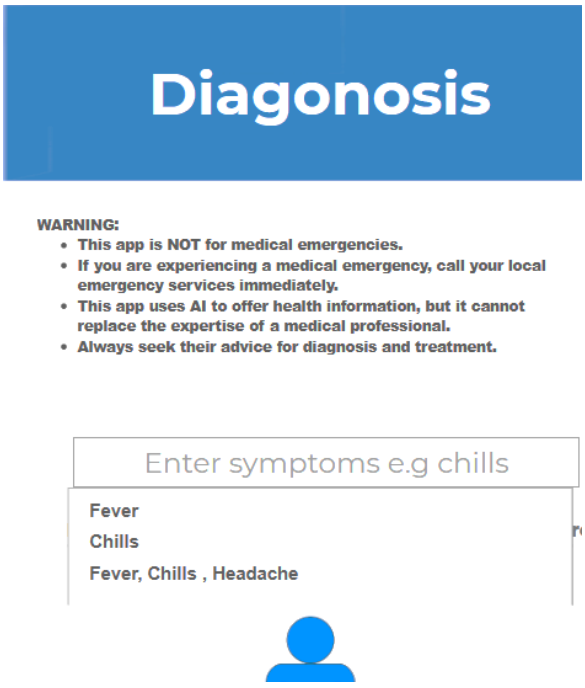

Figure 11: Dropdown Menu with Example Symptoms for New Users

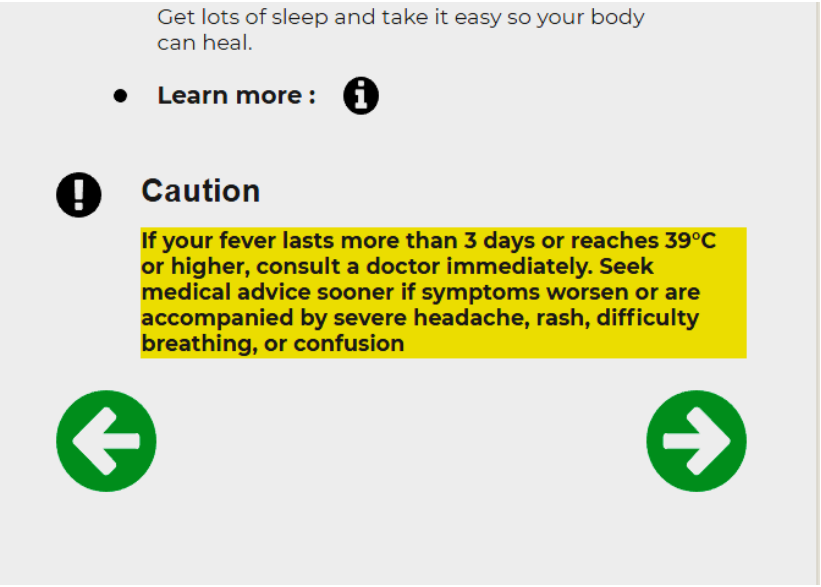

Figure 12: Navigation Arrows Display

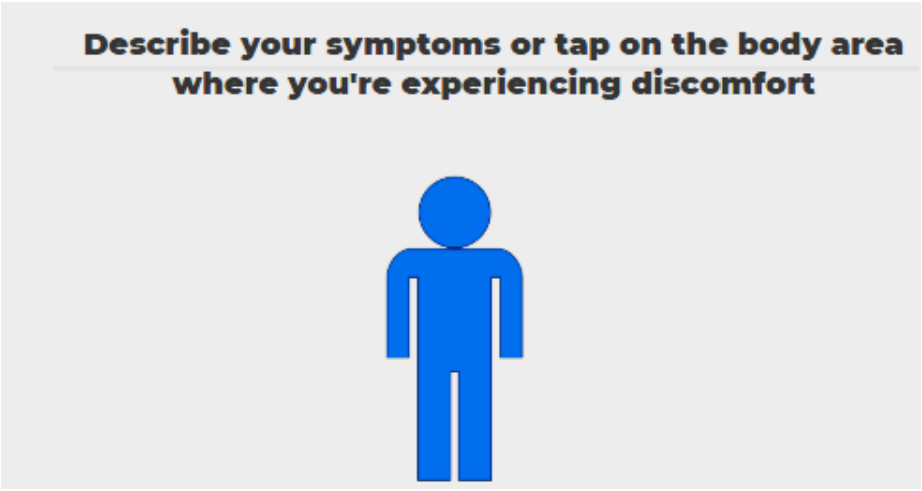

Figure 13: Interactive Diagram Search Feature

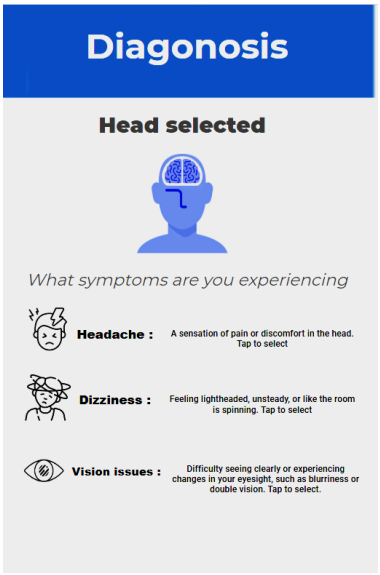

Figure 14: Associated Symptoms for Selected Body Part

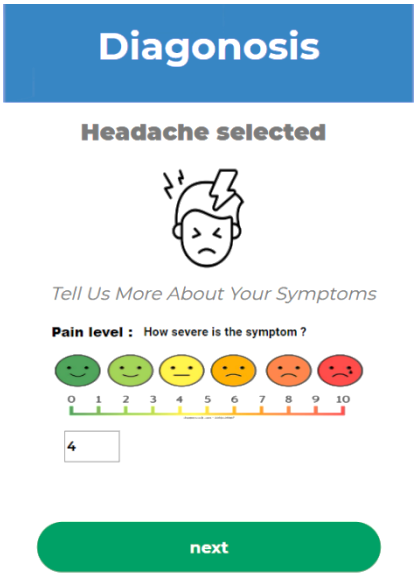

Figure 15: Pain Scale for Selected Symptom

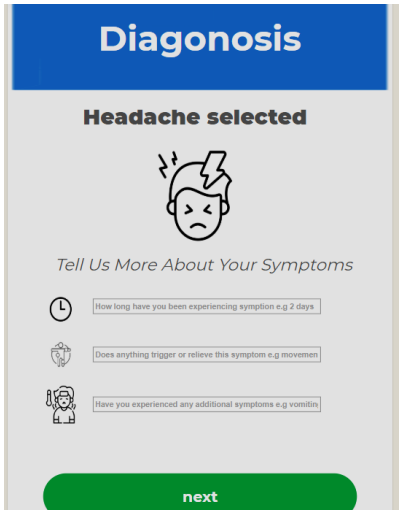

Figure 16: Additional Questions and Prompt for Adding More Symptoms