Marvin - A Chatbot For Medical Record Management

Michael Gamston [*1]

Abstract

This project presents ”Marvin”, a proof-of-concept chatbot designed to assist users in accessing, modifying, or deleting their medical records. Particularly those with who might not be technologically able. The chatbot leverages string parsing, tokenisation, lemmatisation, and bag-of-words representations for user input processing and Support Vector Machines trained on custom datasets to interpret user intents. Marvin is implemented with a graphical user interface developed using PyQt5. Through experimental comparisons and real-world testing, the system demonstrates promising accuracy and usability, although limitations in handling non-standard inputs and low generalisation beyond training data highlight areas for future improvement.

1. Introduction

This project aims to develop a proof of concept tool for a user to retrieve, delete and request changes to their medical records, with the idea that it could help people who are not technologically able. The current method for accessing NHS records requires a user to log into the NHS website or app, or alternative call their GP. For users who are unfamiliar with the internet and have speech or hearing issues this can be a daunting task. This project primary aim is to trying to show that a more accessible approach to tasks like this is possible and would likely help the more vulnerable members of our society.

2. Project Setup

2.1. The Environment

There are two parts of this environment. The first, a user. This user interacts with an interface that is the front facing part of this project. Through this interface the user is able to correspond with the second part of the environment, Marvin, the language processing agent that is used to carry out the request of the user.

2.2. The Agent

The agent is a language model, named Marvin. The agent has been named because it makes the overall experience more personable to the user. This agent uses natural language processing (NLP) techniques such as tokenisation and lemmatisation to preprocess the users input before using traditional machine learning (ML) and string parsing techniques to to classify the users intent.

2.3. Aims

This project aims to try and successfully build a project that can help ease the process of manipulating medicals for those who are not technologically able. Once this is built the project aims to use it to answer the following questions:

- Can an intelligent system improve system accessibility for technologically challenged people?
- How well does the agent respond to non-standard requests?
- How well does the agent compare to other similar agents?

3. Literature Review

This section of the report will detail similar projects, their approaches and their results. Using NLP and traditional ML techniques to do text classification is nothing new. Many projects have detailed there experiments in this field. The present authors research has suggests that support vector machines in combination with bag of words are a popular choice for lightweight NLP problems.

A project by Alanazi et al. outlines the possible uses of machine learning techniques combined with natural language processing to automatically vet referrals, saving medical professionals time. The authors of this paper decide to compare three traditional and two deep learning techniques; support vector machines (SVM), logistic regression (LR) and random forest (RF) along with convolutionary neural networks (CNN) and recurrent neural networks (RNN). They also

UoN [1]Department of Physics and Astronomy, University of Nottingham, UK, NG72RD.

experiment with different word embeddings such as bag of words (BOW), bigram and TF-IDF. The authors write that SVMs with BOG word embeddings managed to score very highly when their training dataset was balanced, however when using an unbalanced dataset RF with TF-IDF representation managed to score better. (Alanazi et al., 2022) Their findings are highly relevant and indicate that and SVM with BOG could be a good choice for this project.

In another paper reviewing SVMs, Li et al. discuss how splitting up multi-class classification tasks to multiple different binary classification tasks can help increase the accuracy of a model. (Li et al., 2006) This is logical, having fewer classes to predict will likely increase accuracy in any model. However, it is not always possible. This projects aim of using ML in NLP is an example of where multiple binary classifiers is not feasible. Although it is worth drawing lessons from as splitting classes where possible could improve the model performance.

Kamath et al. write another comparison between ML techniques in text classification. They compare the same techniques as Alanazi et al. but come to different conclusions. The authors find that LR has the best scores among traditional ML but CNNs provide the best overall score. (Kamath et al., 2018) This contradicts the findings by Alanzi et al. However, it is important to note that Kamath et al. use a custom word embedding and that results will vary between models depending on the format of the word embeddings.

4. Data

4.1. Health Data

This project is a proof of concept and therefore did not require access to a real medical dataset. However, the project would require some form of medical data for the system to manipulate. A dataset titled "Healthcare Dataset" from Kaggle was found to contain an adequate amount of features and data points. This data set contains 40235 unique data points with 15 features per data point. These features are: Name, Age, Gender, Blood Type, Medical Condition, Date of Admission, Doctor, Hospital, Insurance Provider, Billing Amount, Room Number, Admission Type, Discharge Date, Medication and Test Results.

4.2. Training Data

The classifiers this projects uses require a training dataset. Considering the unique and specific nature of the project it would be impossible to find a pre-compiled dataset that would satisfy the projects needs. Because of this, a considerable amount of time has gone into compiling 4 sets of data for this project.

Each dataset created for this project consists of 2 features; input, which is a sentence asking or stating something, and an intent, which is the label that the classifier will associate the input with. Each dataset varies in size but for each intent there are 200 unique inputs. This is still a relatively small amount of data for a ML model to train with. The sets were created by asking ChatGPT to create sentences of varying sizes with various sentiments and contexts based off an example sentence. For instance, a prompt might look like:

> "Create me 50 sentences, of varying lengths from 2 words to 10, using varying sentiments, based off of the sentence: 'Please may I access my medical record'"

These sentences were then checked manually to avoid any that were too unrelated or strange and then added to a CSV and given a correlating intent tag.

Section 7.2 will detail the specific use of each dataset.

5. Methods

5.1. Tokenisation and Lemmatisation

To perform tokenisation and lemmatisation this project will use SpaCy's python API. Tokenisation is used to split a text input up into tokens, each token containing one word. This project also uses tokenisation to get rid of punctuation as it is not important for this classification task. Lemmatisation is the process of reducing words to their base term or lemma. For instance, eating, eaten and eats would then become eat. (Murel & Kavlakoglu, 2023) Lemmatisation is used to reduce model complexity by reducing the size of the feature space. A report by Naji Hussain et al. compares some feature reduction techniques with different machine learning models. Although the project by Hussain et al. is not directly related to this project it does highlight the importance of reducing the feature space to reduce noise, which lemmatisation does.(Naji Hussain et al., 2023)

5.2. Word Embedding

Once tokenisation and lemmatisation have been carried out the resulting text data needs to be transformed into a ML readable data type, this is done with word embedding. Word embedding transforms string data into vectors. There are two popular techniques for this when concerned with a project such as this were the dataset is small and the aim is very specific; mean embedding representation and BOW representation.

Mean embedding representation works by associating each lemmatised token with a pre-trained vector. This vector is unique to that token and contains information that relates to the context of that particular word. The resulting vectors are

then averaged, to gain a final representation. This method is able to keep track of some of the semantics, however any positional information is lost when the average is taken.

BOW representation is quite a simple method, however for small projects sometimes this can be beneficial. BOW represents each section of text with a vector. The size of this vector is described by the size of its observed vocabulary. It simply tracks which tokens appear in the text and counts them, a tokens position in the vector is then updated with the count of that token. (Murel & Kavlakoglu, 2024)

This project will mainly focus on BOW as the related literature in section 3 describes it to the most effective. This is also backed up by experiments detailed in section 6.

5.3. Support Vector Machines

SVM's are very popular choice for simple text classification tasks, they have been used frequently in related projects. Both the literature in section 3 and this projects experiments in section 6 suggest SVMs are the best choice to tackle this sort of problem. SVMs were invented by Vapnik et al. and published in their 1996 paper "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing".

The simplest form of an SVM is the linear SVM, this tries to find a decision boundary that best separates the classes, by maximising the margin size between data clusters. Figure 1 shows a visualisation of this. The grey box represents the margin the algorithm has tried to maximise and the black line in the centre presents the decision boundary. (Vapnik et al., 1996)

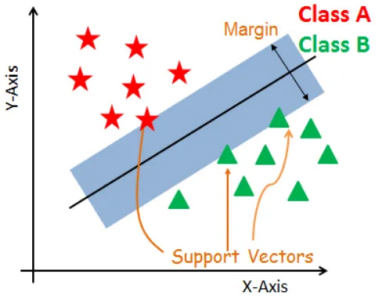

Figure 1. Visualisation of a Linear SVM. From (Panuganti, 2020)

Data is not always linearly separable, especially in the case of text classification where the feature space is often large a quite noisy. SVMs are able to account for this by using kernels. Kernels are able to transform the feature space into a higher dimensionality so that a hyperplane can be found between the class clusters. A visualisation of this process is shown in figure 2.

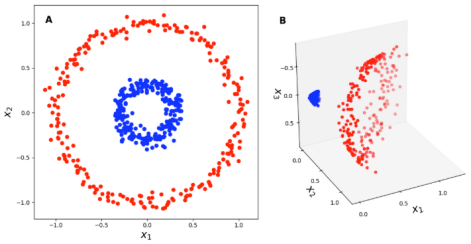

Figure 2. Visualisation of a kernel being applied in low dimensional feature space. From (Gundersen, 2019)

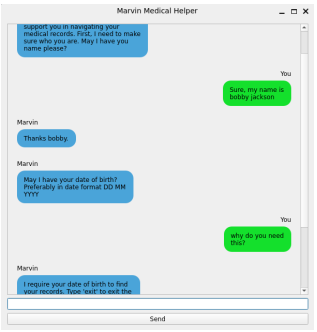

Figure 3. A screen shot of the GUI created with PyQt5

This project will focus mainly on SVMs using the radial bias function (RBF) kernel. This kernel is able to map data to an infinite-dimensional space, and is suggested to the most popular kernel to apply. However, it can become computation expensive due high-dimensionality mapping. (Pilario et al., 2020)

5.4. PyQt

This project uses PyQt5 to develop a graphical user interface (GUI). PyQt5 is a python port of a popular C++ framework for developing cross-platform GUIs. It is used by developers to create lightweight but professional looking applications for any major operating system. Figure 3 shows a screenshot of the GUI created for this project. Section 7.1 will detail the development process.

5.5. Evaluation Metrics

This section will give a brief overview of some of the evaluation metrics used in this project.

Accuracy - This metric measures the percentage of correct predictions out of all predictions. This metric can sometimes be skewed and inaccurate if the data is imbalanced. However, considering all datasets in this project are balanced it is an appropriate metric to use.

$$Accuracy = \frac{Correct\ Pedictions}{All\ Predictions} \tag{1}$$

Precision - This metric displays the percentage of correctly identified positive predictions out of all positive predictions.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

Recall - This metric displays the percentage of positive predictions the model is correctly picking out of all the total possible positives.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3)$$

6. Experiments

The research detailed in section 3 has shown that results can vary between projects. Different word embeddings, tokenisation techniques and feature sizes can cause difference in results, which is why it is important to conduct experiments within the specific problem space.

This project expirments with 5 different traditional ML model types and 2 different word representation types. The models types are: Decision Trees, Naive Bayes, Logistic Regression, Support Vector Machines and Multi-layered Perceptron. These were chosen as they are all mentioned in literature as capable models. The representation types will be mean word embedding and bag of words.

The largest dataset of the 4 described in section 4.2 was used for these experiments. This dataset contains 6 intents: other, question, get record, delete record, update record and change user. With 1200 overall data points.

This dataset was split with 20:80 test to train spilt. The experiments were carried using using 5 fold cross validation on train split. Cross validation is useful as it provides a more accurate representation of the model by averaging the scores over 5 iterations of training. 5-fold cross-validation works by:

1. Split the dataset into 5 equal-sized parts (folds).
2. Use 4 folds to train the model.
3. Use the remaining 1 fold to test the model.
4. Rotate the test fold each time so that every fold is used exactly once for testing.
5. Repeat steps 2, 3 and 4 until each fold has been used as the test fold. Then average the score.

This method provides a more representative performance estimate compared to a single train-test split. A grid search to find optimal parameters was also added to the cross validation. Each algorithm was tested using both mean word representation and bag of words representation.

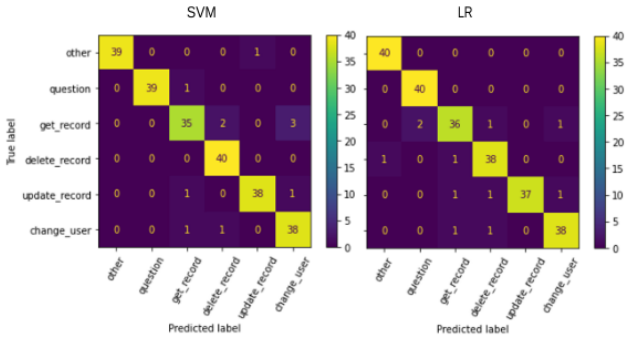

Figure 4. A comparison between the confusion matrices of the LR and SVM with BOW experiments.

The following results were attained by using the test split. Table 1 displays accuracies acquired using mean embedding representation, table 2 displays accuracies acquired using bow of words representation.

	Accuracy	Precision	Recall
DT	0.5	0.51	0.5
NB	0.61	0.61	0.60
LR	0.82	0.82	0.81
SVM	0.88	0.88	0.88
MLP	0.84	0.84	0.83

Table 1. A table displaying results from experiments using mean embedding representation

	Accuracy	Precision	Recall
DT	0.82	0.83	0.82
NB	0.92	0.92	0.93
LR	0.95	0.95	0.95
SVM	0.95	0.95	0.95
MLP	0.93	0.93	0.93

Table 2. A table displaying results from experiments using bag of words representation

Each data point in table 2 is higher than its counter part in table 1, in some cases improving by over 20%, which is significant. These findings along with the research detail in section 3 confirm that BOW is the best for this project.

LR, SVM and MLP all performed well, each achieved a score of 93% or higher. With SVM and LR scoring top by 2%. What was surprising to see was how similar accuracy, precision and recall were for each algorithm. This could explained by that fact the training dataset was quite small and completely balanced. This suggests that the top three algorithms were able to achieve a high level of generalisation with low bias and low variation.

Figure 4 displays a comparison of confusion matrices of

the two best performing models, SVM and LR both with BOW. These matrices show that both the SVM and LR have a relatively even spread of miss-classifications. Given the results in table 2 and figure 4 it would be impossible to select a optimal model. However, the literature more often mentions SVM as the better choice for text classification. For this reason SVM was chosen over LR.

7. Implementation

This next section will detail the design and construction of the agent.

7.1. GUI

As mentioned in section 5.4 the GUI was built using PyQt5. Figure 5 shows an annotated screen of the GUI. There are 5 main parts to GUI, they are:

1. This is the main window, this window contains a centralised widget, that is used to hold the title along with the other widgets mentioned below.
2. This is a scrollable widget with a scroll bar. This scrollable widget contains the chat box widget, which is used to store the stack of messages.
3. This widget is placed outside of the scrollable widget and is used for the user to input a response.
4. This is a button widget, containing the send button for the user to return their input.
5. This is the response widget. When a response comes in, a new widget is created, this widget contains the title of the responder and is filled with the response and a corresponding colour to indicate the responder.

7.2. Marvin

Where possible this project tries to reduce the number of categories each classifier is trained to predict. Spreading the classification over 4 separate models rather than once large one. As mentioned by Li et al. reducing the number categories can help accuracy. (Li et al., 2006)

Marvin uses a knowledge dictionary to track the users progress through the system. This dictionary contains several keys know as states; name, date of birth, intent, handle intent, carry out request.

Each of these states is initially set to 'UNKNOWN'. When Marvin initialises a message is posted requesting the users name. Once the user responds, the users input is tokenised and lemmatised with SpaCy, as detailed in section 5.1. Once this is done a function call is made to get a response from Marvin.

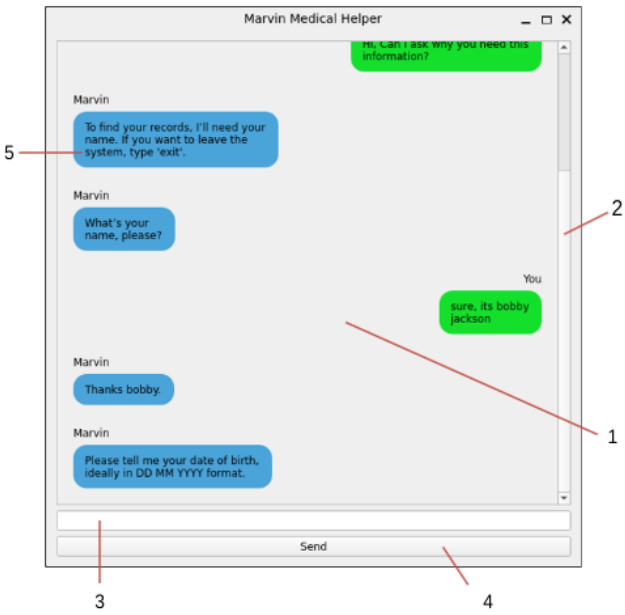

Figure 5. An annotated screenshot of the GUI.

The flow of this function call is laid out in figure 6. Initially a state is found by searching through the knowledge dictionary to find the first key that has an 'UNKNOWN' value. The order of these states is annotated in figure 6.

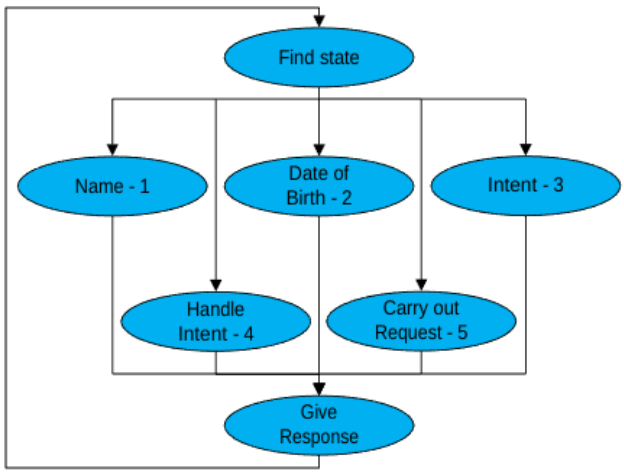

Figure 6. A diagram showing the flow of states Marvin uses.

Once a state is found a function call is made to extract the necessary information from the users input. Marvin also looks for certain key words:

- 'exit' - If this is entered the system will exit.
- 'help' - This keyword will prompt Marvin to respond

with a detailed guide of how a particular state words.

Each of the responses from Marvin are tailored to the individually state that it is responding from. Each state has its own JSON file of corresponding responses. After each state is exited Marvin checks which state will next be entered and responds with a question asking for some relevant information. The next subsections will a give an overview of what happens in each state.

7.3. Name

This state tries to find a name within the users input using string parsing and SpaCy tokens. It utilises an SVM if no name is found, that has been trained to recognise 3 categories; other, question and change user.

- Other is triggered when the users response is seemingly unrelated, this then triggers Marvin to ask for a clarification.
- Question is triggered if the user is asking some form of question, this triggers Marvin to print out a statement on how the specific section works.
- Change user triggers Marvin to clear the users data to make way for a different user.

Once a name has been found, it is checked against the names in the health dataset. If this check is successful then Marvin updates the name key in the knowledge dictionary with the users name.

7.4. Date of Birth

This state runs very similarly to the Name state. It finds the users date of birth within their response. If no date of birth is found it utilised a state specific SVM, but looks for the same categories as the SVM detailed in section 7.3.

Once a date of birth has been found, it checks it against the name that has been found to see if it matches. If it does it adds the users age to the knowledge dictionary and exits the state.

7.5. Intent

This state tries to find what the user wishes to do with their medical data. This state primary consists of the SVM trained in section 6. Using the same categories it finds if the user wants to delete, retrieve or change any data. It also looks for the categories detailed in section 7.3.

If the user also includes their target data points with their intent request then these are found using string parsing and added to the 'Handle Intent' key in the knowledge dictionary. Once the users intent is found it is added to the 'Intent' key in the knowledge dictionary.

7.6. Handle Intent

If no target data points have been identified in the 'Intent' state then this state seeks to find them using string parsing. If no targets are found then it uses another state specific SVM with the same categories detail in section 7.3.

Once the data points are found they are added to the 'Handle Intent' key in the knowledge dictionary.

7.7. Carry Out Request

This is the stage for Marvin to carry out the request. Marvin first asks the user to confirm if they want to continue with this request. Once the user has confirmed the request, it is carried out using all the information gathered previously.

Marvin either, deletes the relevant information from the database, displays it in the chat or asks what the user would like to change each data-point to. Once all changes are received they are added to a JSON file for review by a 'doctor'. Once this is done both 'Intent' and 'Handle Intent' are reset to 'UNKNOWN'.

8. Tests, Results and Questions

This section will review how well the project went and whether or not it was able to answer the questions set out at the start of the report.

8.1. Tests and Results

To test the usability of the system hand written test sets were constructed. Each classifier had its own related test set. Each set contained 10 test example inputs per category.

	Accuracy	Precision	Recall
Name state SVM	0.9	0.91	0.9
D.O.B state SVM	0.9	0.9	0.9
Handle Intent state SVM	0.83	0.82	0.83
Intent state SVM	0.81	0.83	0.81

Table 3. A table displaying results of the final tests

Table 3 displays how well the models have done on previously unseen data. It is clear to see that none of models were able to achieve a score similar to the training in section 6. The accuracy of the intent state SVM has drop by over 10%. This is also reflected in the user experience of Marvin. Most of the time Marvin is able to correctly identify user intents, however there are notable mistakes that happen.

Figure 7 shows the confusion matrices for the name state, D.O.B state and handle intent state test results. It shows that

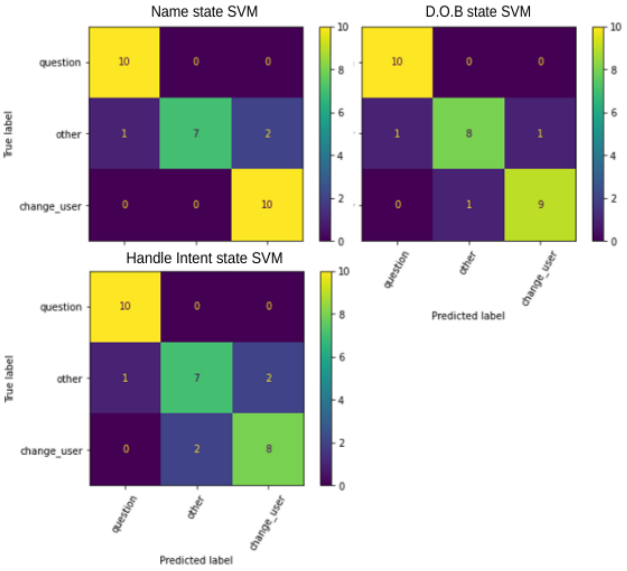

Figure 7. A display showing the name state, D.O.B state and handle intent state test result confusion matrices.

the classifiers are able detect questions very well, but they generally have poorer precision when classing unrelated 'other' data.

Figure 8 shows the confusion matrix for largest SVM, the intent state classifier. It shows that generally it is quite good at predicting the correct category, but not perfect. It shows that, like the other models, it still struggle with the 'other' category.

Overall we can see a drop in the accuracy from the experimental data published in section 6 to test result data displayed in this section. This could be explained by the way the training was compiled. When ChatGPT was promoted to create the training data it was requested to change up the semantics and context of the sentences, however, it still imparts its own style. ChatGPT has its own style of writing, this is well known and generally recognisable even by humans. Its possible the classifiers have learnt this style of writing well and so when they see differing style, such as that in the hand made test data, its unable to as closely match it to the correct categories.

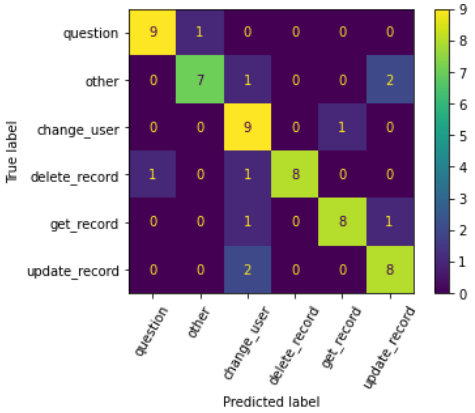

Figure 8. A display showing the test result confusion matrix of the intent state classifier.

Marvin has also been tested manually, by running through certain user scenarios, and has shown to be capable for fulfilling user requests.

8.2. Questions

8.2.1. Can an intelligent system improve system accessibility for technologically challenged people?

This project has shown, with a relatively successful completion of creating an easy to use chatbot, that it is possible to have a system that increases the accessibility of medical record management. If Marvin were used on the website as a pop up or perhaps as an app that a user could access through their mobile it could help solve some of the issues raised in section 1.

8.2.2. How well does the agent respond to non-standard requests?

Currently the agent is able to handle non-standard input, however it is its weakest area. Non-standard input is current handled by the classier by predicting 'other' and a message appearing asking for clarification. Without having have a more complicated model that could understand the semantics of each sentence, such as an LLM, it would be hard to improve on this.

8.2.3. How well does the agent compare to other similar agents?

There are many other similar agents doing comparable tasks on the market currently. For instance, Just Eat have launched an AI assistant where a user is able to make requests to order food from restaurants. The present author has personal experience of this AI, it handles non-standard data in similar fashion to Marvin. For instance, it was asked "what's the weather like today" and it responded with "I'm sorry, I can't help with that. What would you like to order". However it is able to deal with a much broader range of requests, to a much higher degree of accuracy. Just Eat's AI doesn't rely on any keywords and the interaction feels more conversational.

Given more time and a switch to a model with a higher computation ability like a LLM Marvin could be just as a good. It currently falls a bit short of others on the market.

However, Marvin is capable of fulfilling the role it provides. The system can still complete the requests it is given and is competent enough to deal with multiple input styles in an intelligent way.

9. Further Works

Given more time there are a few improvements that could be made. The first is some form of feature extraction and feature reduction techniques. In a paper writen by Nguyen et al. the authors stress the importance of feature extraction in classification models. Suggesting that unnecessary features adding noise to the feature space could decrease the accuracy of the model.(Nguyen & de la Torre, 2010) In theory simplifying the feature space to only include relevant features would increase the variance in the data allowing the SVM to build larger margins between the data clusters.

The project could also be improved by developing a LLM to help find context from the user input. Banerjee et al. writes in a paper that LLMs are able to encode knowledge within themselves unlike traditional ML, saying this could help a NLP model respond with higher accuracy and user understandability. (Banerjee et al., 2023) Having a model that can write it's own response and not have to rely on premade responses would greatly improve the generalisation of the chatbot. It would certainly help the models ability to response to non-standard inputs.

Being able to create a larger dataset of real user requests would also greatly improve generalisation of model, and would address the issues of accuracy disparity discussed in section 8.1.

Finally adding the ability of a user to verbally interact and receive an auditory response from the system would greatly improve its usability for individuals with disabilities. This would further help address the question discussed in section 8.2.1.

10. Conclusion

The development of Marvin shows that a chatbot can make medical record management more user friendly. SVM and BOW provide acceptable accuracy, allowing Marvin to complete a range of different requests. While Marvin falls short compared to commercial solutions in handling non-standard inputs, it proves the concept of an accessible medical chatbot. Further development involving larger datasets, more diverse training examples, feature extraction techniques, and an integration of a large language model could significantly improve its conversational ability and user-friendliness.

References

Alanazi, A. H., Cradock, A., Ryan, J., and Rainford, L. Machine learning and deep learning-based natural language processing for auto-vetting the appropriateness of lumbar spine magnetic resonance imaging referrals. *Informatics in Medicine Unlocked*, 2022.

Banerjee, S., Dunn, P., Conard, S., and Ng, R. Large language modeling and classical ai methods for the future of healthcare. *Journal of Medicine, Surgery, and Public Health*, 2023.

Gundersen, G. Implicit lifting and the kernel trick, 2019. URL https://gregorygundersen.com/blog/2019/12/10/kernel-trick/.

Kamath, C. N., Bukhari, S. S., and Dengel, A. Comparative study between traditional machine learning and deep learning approaches for text classification. In *Proceedings of the ACM Symposium on Document Engineering 2018*. Association for Computing Machinery, 2018.

Li, Y., Bontcheva, K., and Cunningham, H. Adapting svm for natural language learning: A case study involving information extraction. 2006.

Murel, J. and Kavlakoglu, E. What are stemming and lemmatization?, 2023. URL https://www.ibm.com/think/topics/stemming-lemmatization.

Murel, J. and Kavlakoglu, E. What is bag of words?, 2024. URL https://www.ibm.com/think/topics/bag-of-words.

Naji Hussain, A., Abboud, S. A., baki Jumaa, B. A., and Abdullah, M. N. Impact of feature reduction techniques on classification accuracy of machine learning techniques in leg rehabilitation. *Measurement: Sensors*, 2023.

Nguyen, M. H. and de la Torre, F. Optimal feature selection for support vector machines. *Pattern Recognition*, 2010.

Panuganti, J. Support vector machines, 2020. URL https://jyothiffu.medium.com/support-vector-machines-65b4a04d177.

Pilario, K. E., Shafiee, M., Cao, Y., Lao, L., and Yang, S.-H. A review of kernel methods for feature extraction in nonlinear process monitoring. *Processes*, 2020.

Vapnik, V., Golowich, S. E., and Smola, A. Support vector method for function approximation, regression estimation and signal processing. In *Proceedings of the 10th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 1996. MIT Press.