# Developing an Intelligent Chatbot

Group 18: Alexander Vranic* - 100350592, James Hamilton - 100340006

May 30, 2024

**Abstract**

ABSTARACTIONS

## 1 Introduction

In recent years, chatbots have become increasingly popular in a variety of applications, as such, so has the technology surrounding them. It is not our mission to compete with the likes of OpenAI (2024*b*), Microsoft (2024) and GitHub (2024*a*) and their hugely successful Large Language Models (LLMS), GPT-4 OpenAI (2024*a*) and CoPilot GitHub (2024*b*) respectively. At the end of the day, the coursework is a learning experience, not a fully fledged product.

Our solution is a small client side chatbot, integrated into an intuitive graphical user interface and designed to handle prompts around a bespoke context. We utilise modern natural language processing (NLP) techniques, with a knowledge base and inference engine, with machine learning and webscraping, in the name of enabling the user with concurrent information to make informed decisions on their train travel plans.

### 1.1 Background and Motivation

#### 1.1.1 Compulsory Motivators (Assignment Brief)

As specified in the CMP6040/7028 assignment brief Wang (2018), tasks one and two are to implement an intelligent conversational system, designed to "*help their customers in finding the cheapest available ticket for their chosen journey*" covered by section 1.2.1 and "*to improve customer service satisfaction by applying some appropriate AI techniques*" covered by section 1.2.2 respectively. Following implicit suggestions from the course contents, our second task implements a delay prediction model, based on historical data (also provided in the course material), in the form of a KNN regressor (Fix & Hodges (1951)), embedded within the original chatbot system created in task one. In depth coverage of our interpretation of the brief seen in section 1.2.1.

Again, as stated in the brief, we are to provide some kind of user interface. Seeing that with our resources, any web based applications would be limited to local hosting anyway, we have decided to create a stand alone desktop application with a graphical user interface (GUI) to mimic the look and feel of modern chat applications. For further clarification, see section 1.2.2.

#### 1.1.2 Chatbot History

Quite some time before the advanced development of AI chatbots, Alan Turing considered the idea of a hypothetical machines ability to think, proposing a method of benchmarking a machines intelligence, aptly named "*The Imitation Game*" Turing (1950), not to be confused with the Turing Test - being the broader concept of measuring a systems intelligence. In the paper, Turing proposed a system involving three parties: a human interrogator, a human respondent and a machine respondent. The core concept

being that the human interrogator must converse with both responding parties and determine which is the machine. The machine is deemed intelligent if it is not reliably distinguished from the human respondent. Turing states that "*at the end of the century*" - being the year 2000 - "*one will be able to speak of machines thinking without expecting to be contradicted*". Though the timing of his prediction can be argued either way, the concept of a machine being able to hold conversation with a human is now a reality, to the point where as a modern humans, we must be consciously question the *human-made* authenticity of the content and media we consume.

Artificial intelligence, designed to mimic human conversation has come a long way since the days of Turing. From the first chatbot ELIZA Weizenbaum (1966) providing incoherent responses diverged from context, to today's CoPilot GitHub (2024*b*) baring the capability to generate code and explain it in any array of natural languages, the technology has become an integral part of our daily lives.

## 1.2 Aim and Objective

As I'm sure you're aware by this point in the report, we are to employ artificial intelligence techniques, in conjunction with webscraping to achieve the following two tasks (Wang (2018)), with data relevant to the current and/or user-specified time frame(s).

1. Finding the cheapest train ticket
2. Improving Customer Service

The following subsections 1.2.1 and 1.2.2 outline our subjective interpretation of the task one and two respectively.

### 1.2.1 Task 1: Finding the Cheapest Ticket

We've taken it upon ourselves to not just find the cheapest individual ticket, but to find the cheapest combination of tickets for a given journey. Once information is derived and tokenised, we achieve this by scraping the SplitMyFare website Split My Fare (2024).

Seeing that their service does incur a small additional fee, initial intentions were to then compare the results from another source, such as Trainline Trainline (2024), to ensure the cheapest pricing of single ticket journeys. However, it would seem that these websites are intentionally or unintentionally difficult to scrape information from, as such, focus was shifted to the components relevant to the names sake of the module. Evidence for which is covered in sections 1.4 and 4, as well as the actual codebase itself *./webscrape/nationalrail.py* Hamilton & Vranic (2024).

As for how this determines the functionality of the actual chatbot. Our system should prompt the user with a series of reactive questions to derive the following information.

#### *1.2.1.1 Information To Derive*

- Desired starting location
- Desired destination location
- Whether to arrive or depart by a certain time
- User specified time and date of departure or arrival
- Number and type of passengers (adults or children)
- Whether they want a return journey
  - If so, then repeat the above, minus passengers

Said information is to be, processed by a natural language processing algorithm, covered in section 4.2, with the resulting tokens updating the state machine responsible for determining its next prompts, until it reaches a completion state. Conversational control covered in section 4.2.2.

On arrival of a completion state, these tokens are to be formatted into a consistent data format, to be passed to the webscraping module, covered in section **??**. The results are then returned to the user via the GUI, covered in section 4.5.

### 1.2.2 Task 2: Improving Customer Service

Given historical performance data from years 2017 to 2022 for the service from London Liverpool Street to Norwich, we aim to provide a prediction for a *actual* delayed time of arrival for journeys either direction between Norwich and London Liverpool Street, given the journey's current locational and time data for comparison to that of the historical data. With a machine learning algorithm known as K Nearest Neighbours (KNN), first publish in paper Fix & Hodges (1951) and later refined in Cover & Hart (1967).

KNN being a simple yet effective algorithm for regression and classification problems, with applications in pattern recognition, we thought it sufficient for the task at hand. It's to be noted that it is a lazy learning algorithm, so works by memorisation of the training data, rather than learning a discriminative function. Regardless, see our coverage of its implementation in section 4.2.2.

The chatbot aspect is to prompt and process user input in much the same way as task one, with different tokenisation parameters and completion states. Again coverage seen in sections 4.2.2.

#### *1.2.2.1 Information To Derive*

- Current time and date
- Current journey direction
- Current/last know station location
- Current predicted delay time given by service provider

This time the appropriate tokens are to be processing into a consistent data format to be passed to the KNN regressor, from which the results can be displayed to the user through user interface, covered in section 4.5.

## 1.3 Difficulties and Risks

As is the case on any scale of software development, there are always factors that bare the potential to hinder progress. The following sections ..... outline the obstacles we aim to take particular to navigate.

### 1.3.1 Mismatching Operating Systems

With one member using Windows 11 Microsoft (2023), and the other using Fedora Linux The Fedora Project (2024), there is potential for one of us to write or import code that the other cannot execute. Our simple solution is to use only cross platform tools and libraries for development, execution and testing of our codebase. Our choices of which are covered in sections 3.2.

### 1.3.2 Unacceptable Range of Output

#### *1.3.2.1 NLP Tokenisation and Responses*

In developing our solution to task one, we'll be bound to come across issues regarding defining our ruleset's for the NLP state machine. This is to be considered regarding our time management, though through careful planning and testing, the time deficit can be minimised.

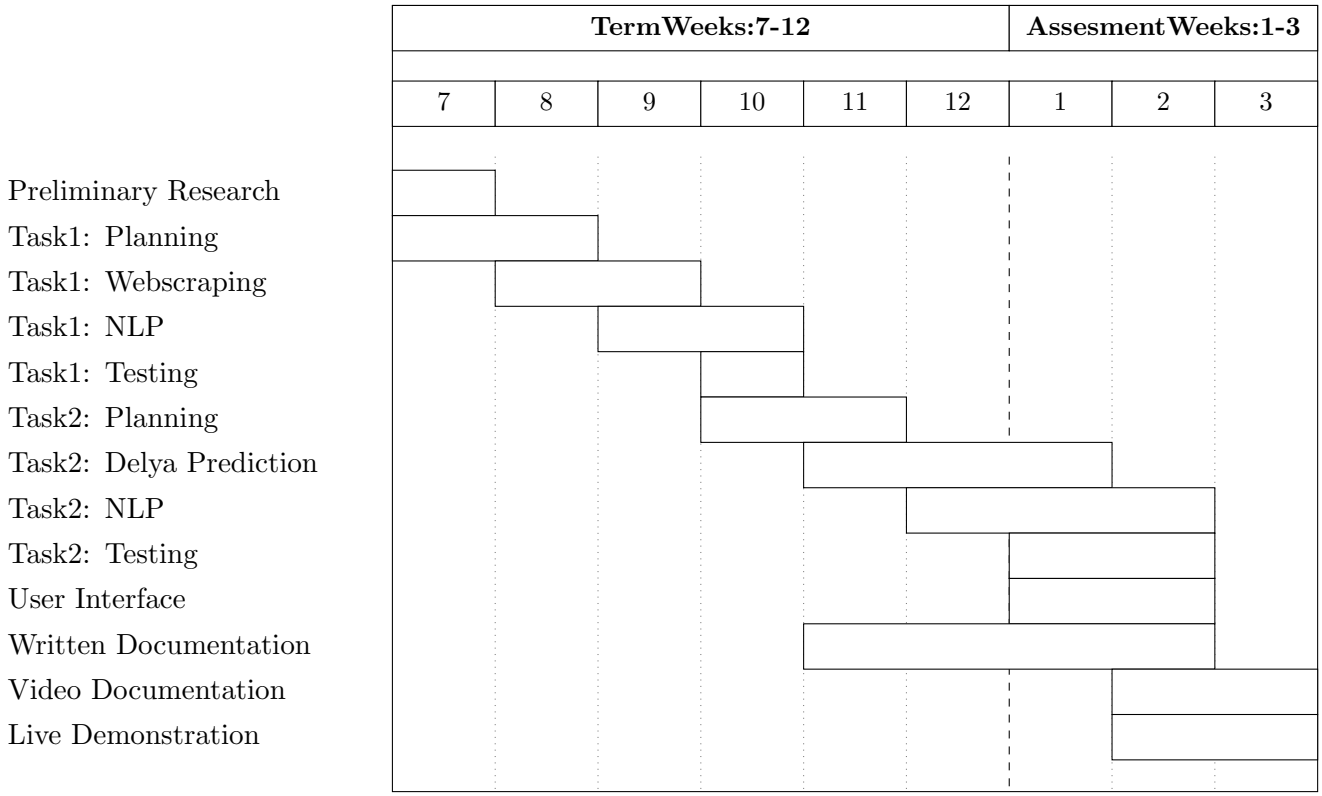
#### *1.3.2.2 KNN Regressor*

Without actually implementing the solution, we are yet to see if the model passes our specifications outlined in section 4.4. If the model is not accurate enough, we may have to consider a more complex model, which could take more time to implement, train and test. All things to consider in our time management covered in section 1.4.

### 1.3.3 Webscraping Deterance Technologies

It's no secret that scraping this information from SplitMyFare's online services is not something they wish to facilitate. Ultimately, between dynamic content (with arbitrary class names and tag ids, generated by frameworks like React Facebook (2024*b*)) and other security measures in place to specifically prevent scraping, time is to be allocated to the development of a robust webscraping module, covered in section 4.3.

## 1.4 Work Plan




# 2 Related Work

Prior to designing our implementation, we conducted some light research into existing chatbot systems and services, to gauge current trends and standards in the field.

## 2.1 Transit Bot (Lekarski (2024)

Transit Bot is a French server-side chatbot service, interacting with it's users via Facebook Messenger (Facebook (2024*a*)), to provide integration with ticket booking services, all through text based natural language conversation, rather than a graphical user interface.

It works in conjunction with private companies to provide an off the shelf package to provide an appealing low commitment (for end users) chatbot service.

It's selling point is a streamlined ticket information and purchasing process. Users don't need to navigate a website or install an application to book their tickets, and with mobile wallet integration, they don't even have to manually provide bank details. It can even provide passenger information, and the means of personalised notifications. All whilst still incorporating the original providers branding and services, at no point hiding said organisations branding.

Being one of the closest examples to our own project, it's a good reference point for what we aim to achieve, albeit on a much smaller scale.

#### 2.1.1 Critical Review

Overall, Assen's achievements with TransitBot are impressive. It's an ethically admirable use of artificial intelligence, allowing company branding to remain uncensored. However, it's reliance on third party services is a key weakness. Reliance on Facebook's API is an inherent security risk, not to mention changes to the API's availability or features could render the service unusable. Overall, heavy inspiration is to be drawn from the rest of the project, albeit on a smaller scale.

### 2.2 ChatBot AI Assistant (Chatbot AI Assist (2024)

ChatBot AI Assistant provides a semi bespoke chatbot service to website owners, giving them accessible means to provide training data and conversation control to their chatbot. It can proactively engage with users through default notification on entry to the website, lead customers through sales and recommended products, automatically generate and qualify sales prospects, provide a user friendly platform to purchase, order or book services, more cost effectively than a human customer service representative.

One of it's primary selling points being that it does not rely on third party LLM services for any of its functionality and consequently it's data is only hosted on their platform. Contrasting to cloud based services like GPT-4 OpenAI (2024*a*).

Though it's purpose and application is starkly different to our own, it's a good reference point for a standalone chatbot service, with a focus on user interaction and malleable conversation control.

#### 2.2.1 Critical Review

ChatBot AI Assistant does a great job of providing companies with accessible means of integrating semi bespoke chatbot systems into their website. With no reliance on third party services, it's increased security and robustness is a key point to consider when creating our own chatbot. Otherwise, we do not aim to incorporate such customisability into our chatbot.

## 3 Methods, Tools and Frameworks

### 3.1 Methods

### 3.2 Languages, Packages, Tools

As mentioned in section 1.3, despite demonstrating on a Windows Microsoft (2023) machine, navigating cross OS development bounds us to cross platform tools and packages. The following outlines our project's dependencies and development environment.

#### 3.2.1 Programming Language: Python (Python Software Foundation (2024*b*)

Our chosen programming language, for all aspects of our system is Python. It's the modern standard for programming in and around AI and machine learning, thanks to it's malleable syntax and vast array ap-

plicable of libraries and packages. Combined with it's user friendly package and environment management tools, explained in the following section 3.2.2, make it an ideal choice for our project.

#### 3.2.2 Environment & Package Managers: Anaconda (Anaconda (2024), Pip (Python Software Foundation (2024*a*)

To be clear, we are primarily using Anaconda as our virtual environment and manager and pip for our packages. It's to be noted, SpaCy's pre trained pipeline had to installed via base python commands. Both abide by our cross platform development policy, whilst providing user friendly interfaces, so we can focus our development time on the actual codebase.

#### 3.2.3 Data Manipulation: Pandas (Pandas (2024)

Before our data can be processed by any one of our models, it must be formatted into the expected data structures. Pandas is a powerful data manipulation library - another industry standard, and consequently a key component in our data processing pipeline.

#### 3.2.4 NLP Package: SpaCy (Explosion AI (2024)

#### 3.2.5 Mathematics & Statistics: Numpy(NumPy (2024)

Though we haven't extensively used its data structures, we utilise Numpy's mathematical functions in key components of our models.

#### 3.2.6 Machine Learning: Scikit-Learn (Scikit Learn (2024)

Our chosen machine learning library, Scikit-Learn, provides us with the means of incorporating machine learning algorithms into our codebase. Prior experience with made it an obvious choice for our project.

#### 3.2.7 Web Scraping Package: Selenium(Selenium (2024)

Selenium allows us to scrape dynamically generated websites (i.e. SplitMyFare (Split My Fare (2024)) and Trainline (Trainline (2024))) via a headless browser emulation, in our language of choice. It comes integrated with measures, like custom user agents and proxies, to bypass anti-scraping measures.

### 3.3 Development Framework

## 4 Design

The following outlines and depicts the design of our system architecture and how key components interface with each-other.

You'll notice, seeing that Python's objects don't have public/private/protected restrictions on attributes and methods, such restrictions are not denoted in any subsequent UML.

### 4.1 Custom Types

As powerful as Python's (Python Software Foundation (2024*b*)) dynamic typing is, it can be a double edged sword. Especially when you consider the number of parameters components in our system have to pass between each other. As such, instead of passing a series of parameters and returning tuples (that are uneditable unless unpacked), we've defined a series of custom types to keep our codebase clean and maintainable. Their attributes and relations are as depicted in the following UML.

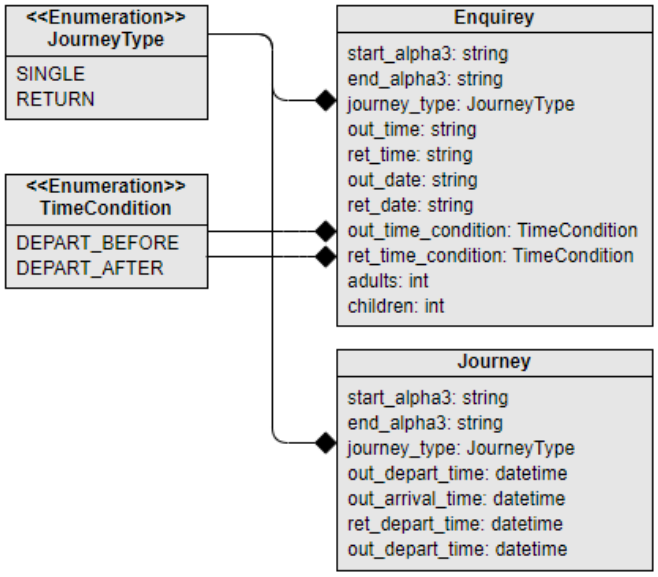



Figure 1: Custom Types UML (Visual Paradigm (2024))

## 4.2 Natural Language Processing (NLP)

### 4.2.1 Custom Named Entity Ruler (NER)

In addition to SpaCy's (Explosion AI (2024)) pre trained pipeline, we've injected our custom NER, into SpaCy's CNN/CPU pipeline, before it's default one, so our custom component can derive the tokens it needs, without interference. Our custom NER primarily involves a pattern set, to match station alpha3 codes to their names in natural language.

Using SpaCy's

### 4.2.2 Dialogue Flow Engine

The purpose of the dialogue flow engine, is to control the flow of conversation between the user and the chatbot, whilst interfacing with the NPL to derive consistently formatted data.

We've implemented it a state machine, dependant on it's current state compared to the data derived with the NPL. It has a default starting state, questioning the user to retrieve as much information as viably possible from the first prompt, after which, consequent states question the user to provide information for the remaining parameters, until the sufficient data is acquired to reach a completion state, from which point, it passes a curated Enquiry object (section4.1) over to the webscraping module.

## References

Anaconda (2024), 'Anaconda', https://www.anaconda.com.

Chatbot AI Assist (2024), 'Chatbot ai assist', https://www.chatbot.com.

Cover, T. & Hart, P. (1967), 'Nearest neighbor pattern classification', *IEEE transactions on information theory* **13**(1), 21–27.

Explosion AI (2024), 'spacy', https://spacy.io.

Facebook (2024*a*), 'Facebook messenger', https://www.facebook.com/messenger/.

Facebook (2024*b*), 'React', https://react.dev.

Fix, E. & Hodges, J. L. (1951), 'Discriminatory analysis, nonparametric discrimination'.

GitHub (2024*a*), 'Github', https://github.com.

GitHub (2024*b*), 'Github copilot', https://copilot.github.com.

Hamilton, J. & Vranic, A. (2024), 'Implementation', https://github.com/j-laze/Ai-CW2-Chatbot.

Lekarski, A. (2024), 'Transit bot', https://www.transit.bot.

Microsoft (2023), 'Windows 11', https://www.microsoft.com/windows.

Microsoft (2024), 'Microsoft', https://www.microsoft.com.

NumPy (2024), 'Numpy', https://numpy.org.

OpenAI (2024*a*), 'Gpt-4', https://openai.com/gpt-4.

OpenAI (2024*b*), 'Openai', https://openai.com.

Pandas (2024), 'Pandas', https://pandas.pydata.org.

Python Software Foundation (2024*a*), 'pip pypi', https://pypi.org/project/pip/.

Python Software Foundation (2024*b*), 'Python', https://www.python.org.

Scikit Learn (2024), 'Scikit learn', https://scikit-learn.org.

Selenium (2024), 'Selenium', https://www.selenium.dev.

Split My Fare (2024), 'Split my fare', https://www.splitmyfare.com.

The Fedora Project (2024), 'Fedora linux', https://fedoraproject.org.

Trainline (2024), 'Trainline', https://www.thetrainline.com.

Turing, A. (1950), 'Computing machinery and intelligence', https://www.cs.colostate.edu/~howe/cs440/csroo/yr2015fa/more_assignments/turing.pdf.

Visual Paradigm (2024), 'Visual paradigm', https://www.visual-paradigm.com.

Wang, W. (2018), 'Artificial intelligence modules (cmp6040, 7028) coursework specification'.

Weizenbaum, J. (1966), 'Eliza—a computer program for the study of natural language communication between man and machine', https://dl.acm.org/doi/pdf/10.1145/365153.365168.