A Project Report

on

# Chubb Conversational AI

BY

**SAI PANDA, ADIT DANEWA, KANISHK YADAV, MOHAK KAPIL**
**2020A7PS0080H, 2019B5A71378H, 2019B2A71452H, 2019B3A70680H**

Under the supervision of

**PROF. ARUNA MALAPATI**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF CS F376: DESIGN PROJECT**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**
**HYDERABAD CAMPUS**
**(JUNE 2023)**

# *Acknowledgements*

We would like to express our sincere gratitude to our faculty mentor, Prof. Aruna Malapati as well as our industry mentors from Chubb, Mr. Thiyaneswaran M and Mr. Pushpendra Kumar for giving us the oppurtunity to work on this project and for their valuable guidance and support throughout this project. Without their important inputs and teachings about the various methods, Conversational AI concepts, this report would not have reached the stage of finality it has.
We thank you from the bottom of our hearts for helping us out whenever we were in doubt.

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**
**HYDERABAD CAMPUS**

# Certificate

This is to certify that the project report entitled, "*Chubb Conversational AI*" and submitted by Sai Prasanna Panda ID No. 2020A7PS0080H in partial fulfillment of the requirements of CS F376, Design Project Course, embodies the work done by him under my supervision and guidance.

**Date:**

**(PROF. ARUNA MALAPATI)**
BITS-Pilani, Hyderabad Campus

## Abstract Sheet
## BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI - (RAJASTHAN) HYDERABAD CAMPUS

## Deisgn Project

**Title:** Chubb Conversational AI

**Name of students:** Sai Panda, Adit Danewa, Kanishk Yadav, Mohak Kapil
**ID Numbers:** 2020A7PS0080H, 2019B5A71378H, 2019B2A71452H, 2019B3A70680H

| **Name of Industry Mentors** | **Designation** |
|---|---|
| Mr. Thiyaneswaran M, Mr. Pushpendra Kumar | Chubb |

| **Name of Faculty Mentor** | **Designation** |
|---|---|
| Prof. Aruna Malapati | Associate Professor, Dept. of CSIS |

**Keywords:** Chatbot, Conversational AI, Knowledge Graph

**Project Areas:** Machine Learning, Deep Learning, Natural Language Processing(NLP), Supervised Learning, Sentence Transformers, Few-shot Learning, Analyzing Research Papers, Pretrained Models

### Abstract

The project aims to build a conversational AI platform in the form of a chatbot that can help internal and external stakeholders to retrieve semantically meaningful and accurate responses based on the free text queries asked by stakeholders. The chatbot should support conversational flow, semantically meaningful fallback and should be able to query a knowledge graph database to retrieve information if requested by the user.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Natural Language Processing

Natural language processing, which evolved from computational linguistics, uses methods from various disciplines, such as computer science, artificial intelligence, linguistics, and data science, to enable computers to understand human language in both written and verbal forms. While computational linguistics has more of a focus on aspects of language, natural language processing emphasizes its use of machine learning and deep learning techniques to complete tasks, like language translation or question answering.

NLP has two main components – Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLU uses syntactic and semantic analysis of text and speech to determine the meaning of a sentence. Syntax refers to the grammatical structure of a sentence, while semantics alludes to its intended meaning. NLU also establishes a relevant data structure which specifies the relationships between words and phrases. On the other hand, NLG is the process of producing a human language text response based on some data input. This text can also be converted into a speech format through text-to-speech services.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:[4]

- Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.

- Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies 'make' as a verb in 'I can make a paper plane,' and as a noun in 'What make of car do you own?'
- Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determines the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).
- Named entity recognition, or NEM, identifies words or phrases as useful entities. NEM identifies 'Kentucky' as a location or 'Fred' as a man's name.
- Co-reference resolution is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).
- Sentiment analysis attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text
- Natural language generation is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

## 1.2 Conversational AI

Conversational artificial intelligence (AI) refers to technologies, like chatbots or virtual agents, which users can talk to. They use large volumes of data, machine learning, and natural language processing to help imitate human interactions, recognizing speech and text inputs and translating their meanings across various languages. [12]

### 1.2.1 Benefits of Conversational AI

- Reduce costs, and increase productivity and operational efficiency through automation: For example, conversational AI can automate tasks that are currently performed by humans and thereby reduce human errors and cut costs.
- Deliver better customer experience, achieve higher customer engagement and satisfaction: For example, conversational AI can provide a more personalized and engaging experience

by remembering customer preferences and helping customers 24/7 when no human agents are around.

- Improve scalability of operations: Conversational AI is also very scalable as adding infrastructure to support conversational AI is cheaper and faster than the hiring and on-boarding process for new employees. This is especially helpful when products expand to new geographical markets or during unexpected short-term spikes in demand, such as during holiday seasons.

### 1.2.2 Examples of Conversational AI

- Chabots: often used in customer service applications to answer questions and provide support
- Virtual assistants: often voice-activated and can be used on mobile devices and smart speakers
- Text-to-speech software: used to create audiobooks, or generate spoken directions
- Speech recognition software: used to transcribe lectures, create transcripts of phone calls, or generate automatic captions for videos

## 1.3 Chatbots

A chatbot is a computer program that uses artificial intelligence (AI) and natural language processing (NLP) to understand customer questions and automate responses to them, simulating human conversation.

Historically, chatbots were text-based, and programmed to reply to a limited set of simple queries with answers that had been pre-written by the chatbot's developers. They operated like an interactive FAQ, and while they worked well for those specific questions and answers on which they had been trained, they failed when presented with a complex question or one that hadn't been predicted by the developers.

Over time, chatbots have integrated more rules and natural language processing, so end users can experience them in a conversational way. In fact, the latest types of chatbots are contextually aware and able to learn as they're exposed to more and more human language.

Today's AI chatbots use natural language understanding (NLU) to discern the user's need. Then they use advanced AI tools to determine what the user is trying to accomplish. These technologies rely on machine learning and deep learning—elements of AI, with some nuanced differences—to develop an increasingly granular knowledge base of questions and responses that are based on user interactions. This improves their ability to predict user needs accurately and respond correctly over time.[11]

### 1.3.1 Common Chatbot Terms

- Action: A single step that a bot takes in a conversation (e.g. calling an API or sending a response back to the user).[6]
- Entity: Keywords that can be extracted from a user message. For example: a telephone number, a person's name, a location, the name of a product
- Fallback: Fallback is an error message that is triggered when the bot doesn't recognize the user's input.[8]
- FAQs: Frequently asked questions (FAQs) are common questions that your users ask. In the context of building an assistant, this typically means the user sends a message and the assistant send a response without needing to consider the context of the conversation
- Happy / Unhappy Paths: Terms used to describe whether the user's input is expected or unexpected. If your assistant asks a user for some information and the user provides it, we call that a happy path. Unhappy paths are all possible edge cases. For example, the user refusing to give the requested input, changing the topic of conversation, or correcting something they said earlier.
- Intent: In a given user message, the thing that a user is trying to convey or accomplish (e,g., greeting, specifying a location).
- Knowledge Base / Knowledge Graph: A queryable database that represents complex relationships and hierarchies between objects.
- NLU: Natural Language Understanding (NLU) deals with parsing and understanding human language into a structured format.
- Story: Training data format for the dialogue model, consisting of a conversation between a user and a bot. The user's messages are represented as annotated intents and entities, and the bot's responses are represented as a sequence of actions.
- Utterance: Anything that the user says is an utterance.

## 1.4 The Transformer Model

A Transformer [9] is a type of neural network that has applications primarily in Natural Language Processing (NLP) and Computer Vision (CV). For example, it can be used to translate text, write poems or generate code. Prior to the introduction of Transformers, sequential data was mainly processed through Recurrent Neural Networks (RNN). RNNs were difficult to train and their gradients would often vanish or explode. They also didn't support parallel processing, i.e., they processed one word at a time which limited the amount of data they could be trained on.

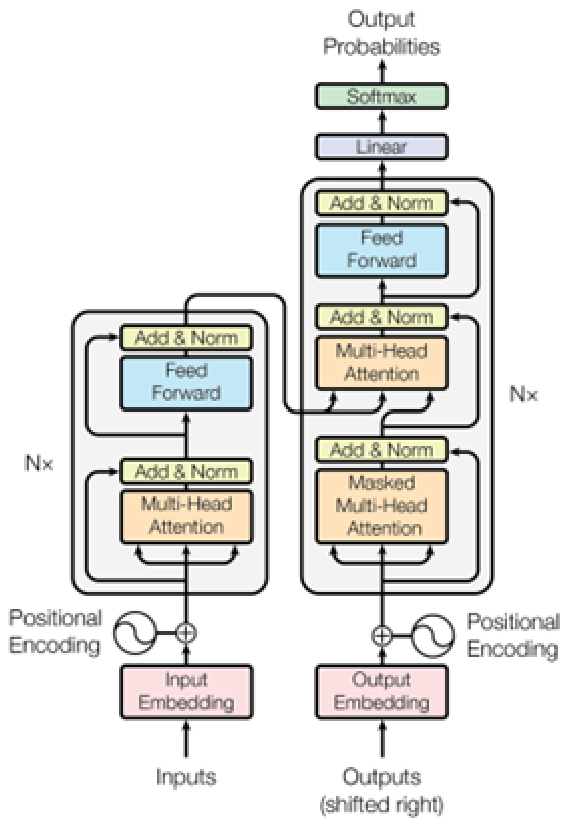


FIGURE 1.1: The Transformer Architecture (Vaswani, A., et al.: Attention is all you need)

Transformers were introduced in 2017 by a team of Google researchers in the paper "Attention is All You Need". Unlike RNNs, they could be efficiently parallelized. There were three main concepts behind Transformers: positional encodings, attention and self-attention. Since they process all the words at once instead of processing them sequentially, Transformers store additional information regarding the position of each word by using sine and cosine functions of varying frequencies. This allowed for the parallelization of the tasks. Attention is a mechanism that allows a model to focus on all the words even while translating one of them. This is useful because some languages have different forms of the same word (for example, a word could be gender-neutral in one language but feminine/masculine in another) whose use depends on other words in the sentence. Attention works by assigning weights to all the words while processing each one of them. The model learns which words to attend to from the training data. However, it was introduced in 2015 and used alongside RNNs in its original form so it was not the major focus of this paper. Self-attention was the most impactful concept introduced in this paper. It allows a neural network to understand words in the context of the other words around them. By understanding the underlying pattern, rules of grammar etc. of each language, Transformers can have a wide range of applications rather than just text translation. There are several pre-trained Transformer models such as BERT by Google and GPT-3 by OpenAI which can be downloaded directly and used.

Transformer models can be used for various components of a Chatbot, such as intent classification, entity extraction, chatbot policy and response generation.

# Chapter 2

# Building the Chatbot

## 2.1 Problem Statement

The goal of this project is to build a conversational AI platform in the form of a chatbot that can help internal and external stakeholders to retrieve semantically meaningful and accurate responses based on the free text queries asked by stakeholders.

We identified the following types of responses that our chatbot should support:

1. Responding to FAQs: If the user asks a frequently asked question, the AI bot needs to respond with a predetermined answer without requiring any additional context. FAQs will be asked from relation with Insurance Domain.
   Example:
   User: What is meant by term insurance?
   Bot: Term life insurance or term assurance is life insurance that provides coverage at a fixed rate of payments for a limited period of time, the relevant term.

2. Responding to user on correct intent classification: If the utterance provided by the user is classified to one of the intents with a confidence score above a certain threshold, the bot should reply appropriately with a response. This response should be randomly chosen from a number of responses all of which convey the same meaning, so that the conversation does not become monotonous.

3. Transactional responses: The user could query for information present in the database and the bot should be able to extract the required entities from the utterance and retrieve the information from the database and display it to the user.
   Example:
   User: what is the status of claim number `<XXXXX>`?

Bot: Please find below status of claim number `<XXXXX>`
Claim Number: `<XXXXX>`
Claim Status: Settled
Insured Name: `<XXXXX>`
Insured Address: `<XXXXX>`

4. Conversational flow: It is possible that a query may span several turns in the conversation as the bot identifies the required entities and prompts the user to enter them. The bot should retain context from the previous messages until it acquires receives all the entities or the user wishes to query something else.

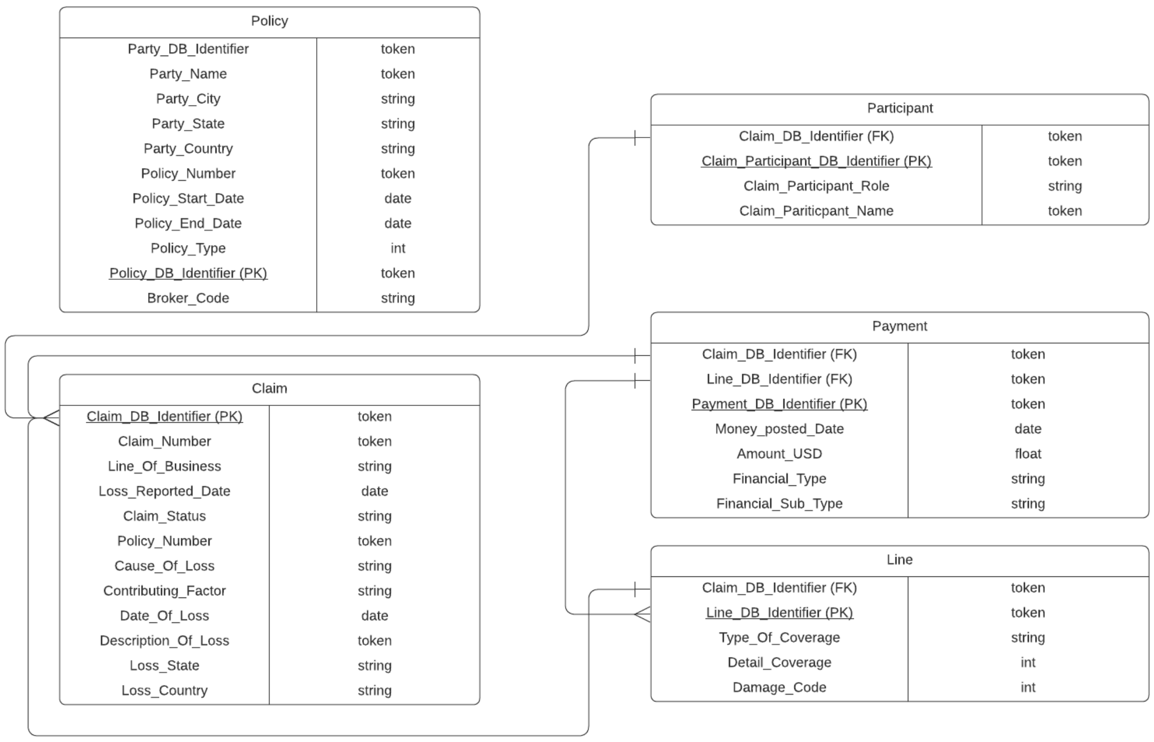
5. Semantically meaningful fallback: If the predicted intent is below the required threshold, the bot should provide additional semantically relevant response along with the default fallback message.
   Example:
   User: what is meant by insurance?
   Bot: I am unable to understand your query. (Fallback response)

   Suggestions: (Semantic responses)
   Are you looking into term insurance?
   Are you trying for personal insurance?
   Are you querying for life insurance?

## 2.2 Dataset

We were provided with five `.csv` files containing policy-related information. The following page contains the entity relationship diagram of the database.

Additionally, we were also provided with a `.doc` file containing approximately 90 Frequently Asked Questions (FAQs) along with their answers.



| Policy | |
|---|---|
| Party_DB_Identifier | token |
| Party_Name | token |
| Party_City | string |
| Party_State | string |
| Party_Country | string |
| Policy_Number | token |
| Policy_Start_Date | date |
| Policy_End_Date | date |
| Policy_Type | int |
| Policy_DB_Identifier (PK) | token |
| Broker_Code | string |

| Participant | |
|---|---|
| Claim_DB_Identifier (FK) | token |
| Claim_Participant_DB_Identifier (PK) | token |
| Claim_Participant_Role | string |
| Claim_Pariticpant_Name | token |

| Payment | |
|---|---|
| Claim_DB_Identifier (FK) | token |
| Line_DB_Identifier (FK) | token |
| Payment_DB_Identifier (PK) | token |
| Money_posted_Date | date |
| Amount_USD | float |
| Financial_Type | string |
| Financial_Sub_Type | string |

| Claim | |
|---|---|
| Claim_DB_Identifier (PK) | token |
| Claim_Number | token |
| Line_Of_Business | string |
| Loss_Reported_Date | date |
| Claim_Status | string |
| Policy_Number | token |
| Cause_Of_Loss | string |
| Contributing_Factor | string |
| Date_Of_Loss | date |
| Description_Of_Loss | token |
| Loss_State | string |
| Loss_Country | string |

| Line | |
|---|---|
| Claim_DB_Identifier (FK) | token |
| Line_DB_Identifier (PK) | token |
| Type_Of_Coverage | string |
| Detail_Coverage | int |
| Damage_Code | int |

FIGURE 2.1: Entity Relationship Diagram of the Database

## 2.3 Initial Chatbot: Rasa

### 2.3.1 Introduction

The first chatbot that we created used the Rasa framework. Rasa is an open source machine learning framework for building AI assistants and chatbots.

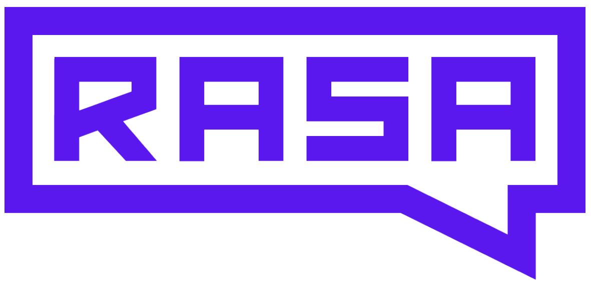


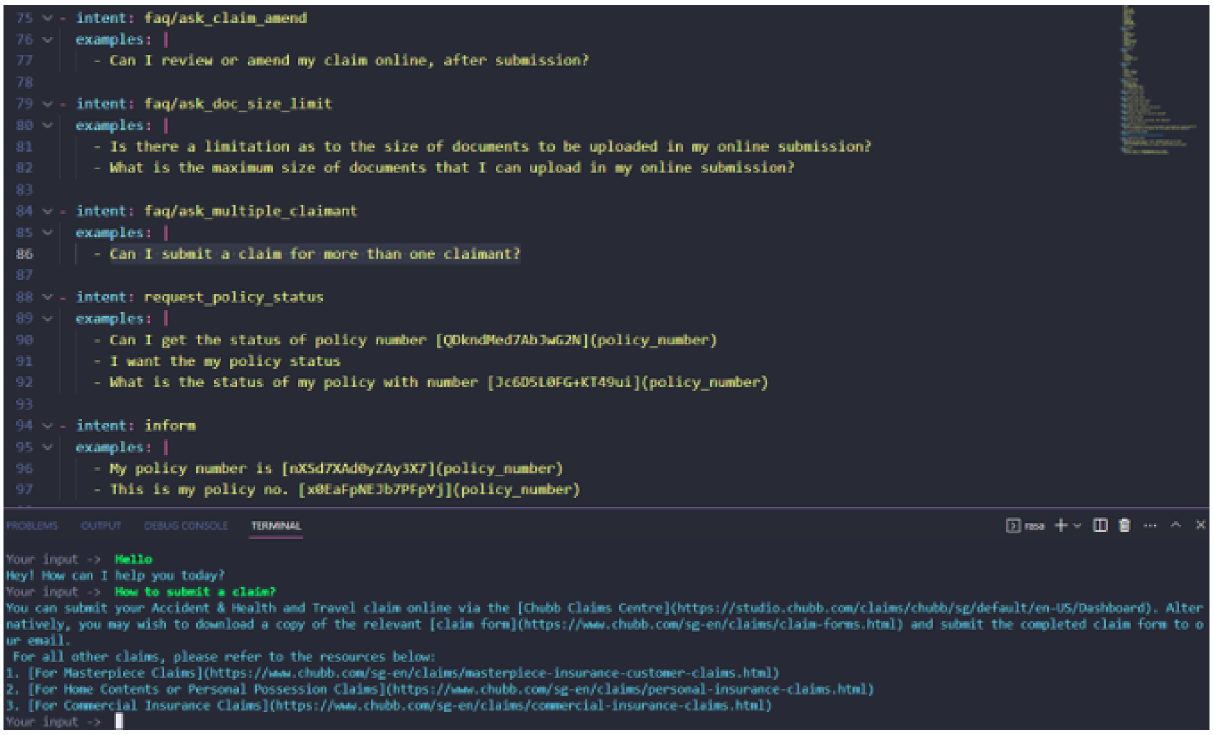
FIGURE 2.2: Rasa Open Source

We used a small number of FAQs and included one transactional query. We hand-crafted the required intents, responses and stories for them and trained the chatbot.

### 2.3.2 Transformer Models

The Rasa chatbot used the following transformer models:

- DIET: DIET is a multi-task transformer architecture that handles both intent classification and entity recognition together. It provides the ability to plug and play various pre-trained embeddings like BERT, GloVe, ConveRT, and so on. It is also lightweight and fast to train.[5]
- TED: TED is a transformer architecture used in combination with rule based and memoization policies to predict the next response of the chatbot.[10]



```
- intent: faq/ask_claim_amend
  examples: |
    - Can I review or amend my claim online, after submission?

- intent: faq/ask_doc_size_limit
  examples: |
    - Is there a limitation as to the size of documents to be uploaded in my online submission?
    - What is the maximum size of documents that I can upload in my online submission?

- intent: faq/ask_multiple_claimant
  examples: |
    - Can I submit a claim for more than one claimant?

- intent: request_policy_status
  examples: |
    - Can I get the status of policy number [QDkndMed7AbJwG2N](policy_number)
    - I want the my policy status
    - What is the status of my policy with number [Jc6D5L0FG+KT49ui](policy_number)

- intent: inform
  examples: |
    - My policy number is [nXSd7XAd0yZAy3X7](policy_number)
    - This is my policy no. [x0EaFpNEJb7PFpYj](policy_number)
```

```
Your input ->  Hello
Hey! How can I help you today?
Your input ->  How to submit a claim?
You can submit your Accident & Health and Travel claim online via the [Chubb Claims Centre](https://studio.chubb.com/claims/chubb/sg/default/en-US/Dashboard). Alter
natively, you may wish to download a copy of the relevant [claim form](https://www.chubb.com/sg-en/claims/claim-forms.html) and submit the completed claim form to o
ur email.
 For all other claims, please refer to the resources below:
1. [For Masterpiece Claims](https://www.chubb.com/sg-en/claims/masterpiece-insurance-customer-claims.html)
2. [For Home Contents or Personal Possession Claims](https://www.chubb.com/sg-en/claims/personal-insurance-claims.html)
3. [For Commercial Insurance Claims](https://www.chubb.com/sg-en/claims/commercial-insurance-claims.html)
Your input ->
```

Figure 2.3: The Rasa Chatbot

### 2.3.3 Drawbacks

The dependence of our initial chatbot on the Rasa framework made it unfit for scaling and it could have led to possible licensing issues in the future as well. It was also rule-based which was something we felt we could improve upon, to make our chatbot more flexible and dynamic. On suggestion of the mentors, we decided to build another chatbot without using the Rasa framework which is described in the next section.

## 2.4 Final Chatbot

Our final chatbot has the following features:

- SetFit for intent classification
- Regex-based entity extraction
- Custom handling of flow-based intents
- Semantically meaningful fallback using sbert
- Gradio interface

### 2.4.1 Intent Classification

#### Introduction

Every utterance from the user has to be classified into one of the intents as the first step of determining the response of the chatbot. This task is handled by our intent classification model. Along with classifying the utterance into an intent category, the model also needs to generate a confidence score for the classification in order to enable fallback handling.

#### Challenges and Possible Approaches

We did not have any real-world conversation dataset between customer support executives and stakeholders, hence we had to synthesize our own data to train whatever model we decided to use. We had the following alternatives which we did not end up using:

- Supervised learning models: These models required large amounts of labelled data to perform well, and since we were synthesizing our data using such a model would require considerable effort in data generation to obtain good performance.
- Zero-shot learning models: On doing some literature review, we came across zero-shot learning models which did not require any training data for intent classification. However, they required meaningful labels for each of the intents which became a difficult task since a lot of the intents had overlapping keywords. We did a comparative study of the `bart-large-mnli` zero-shot classification model with the SetFit model that we finally ended up using, and we observed that in most cases the confidence scores of the zero-shot classification model were lower:

| **Utterance** | **Label** | **ZSC Score** | **SetFit Score** |
|---|---|---|---|
| How can I submit a claim? | submit claim | 0.5335 | 0.5715 |
| After submission, can I review or amend my claim online? | claim review | 0.3794 | 0.6239 |
| I want the status of policy number P112343255245? | request policy status | 0.7941 | 0.8625 |
| What is the maximum size of documents that I can upload in my online submission? | document size | 0.8521 | 0.7961 |
| What are the supporting documents required for travel claim submission? | supporting documents | 0.4890 | 0.6439 |
| In the event of a death claim, who is entitled to receive the insurance benefit under my policy? | death claim | 0.5132 | 0.6240 |
| Show me all the policies from Belgium | request policy by location | 0.5326 | 0.7926 |

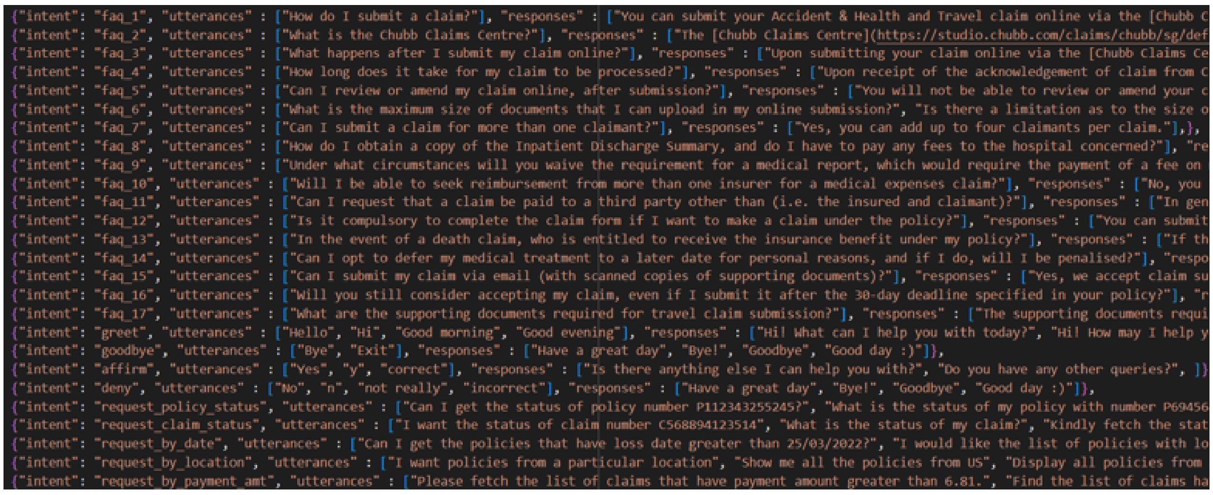
### About SetFit

SetFit is an efficient framework for few-shot fine-tuning of Sentence Transformers. It achieves high accuracy with little labeled data - for example, with only 8 labeled examples per class on the Customer Reviews (CR) sentiment dataset, SetFit is competitive with fine-tuning RoBERTa Large on the full training set of 3k examples.[7]

FIGURE 2.4: SetFit: Efficient Few-Shot Learning Without Prompts

We created a sample dataset using all the FAQs and 5 different flow-based intents that required transactional responses and fine-tuned the pre-trained `paraphrase-mpnet-base-v2` SetFit model on it.

We also generated the confidence scores for each classification of our fine-tuned SetFit model and used it to handle fallback responses.



```
{"intent": "faq_1", "utterances" : ["How do I submit a claim?"], "responses" : ["You can submit your Accident & Health and Travel claim online via the [Chubb C
{"intent": "faq_2", "utterances" : ["What is the Chubb Claims Centre?"], "responses" : ["The [Chubb Claims Centre](https://studio.chubb.com/claims/chubb/sg/def
{"intent": "faq_3", "utterances" : ["What happens after I submit my claim online?"], "responses" : ["Upon submitting your claim online via the [Chubb Claims Ce
{"intent": "faq_4", "utterances" : ["How long does it take for my claim to be processed?"], "responses" : ["Upon receipt of the acknowledgement of claim from C
{"intent": "faq_5", "utterances" : ["Can I review or amend my claim online, after submission?"], "responses" : ["You will not be able to review or amend your c
{"intent": "faq_6", "utterances" : ["What is the maximum size of documents that I can upload in my online submission?", "Is there a limitation as to the size o
{"intent": "faq_7", "utterances" : ["Can I submit a claim for more than one claimant?"], "responses" : ["Yes, you can add up to four claimants per claim."],},
{"intent": "faq_8", "utterances" : ["How do I obtain a copy of the Inpatient Discharge Summary, and do I have to pay any fees to the hospital concerned?"], "re
{"intent": "faq_9", "utterances" : ["Under what circumstances will you waive the requirement for a medical report, which would require the payment of a fee on
{"intent": "faq_10", "utterances" : ["Will I be able to seek reimbursement from more than one insurer for a medical expenses claim?"], "responses" : ["No, you
{"intent": "faq_11", "utterances" : ["Can I request that a claim be paid to a third party other than (i.e. the insured and claimant)?"], "responses" : ["In gen
{"intent": "faq_12", "utterances" : ["Is it compulsory to complete the claim form if I want to make a claim under the policy?"], "responses" : ["You can submit
{"intent": "faq_13", "utterances" : ["In the event of a death claim, who is entitled to receive the insurance benefit under my policy?"], "responses" : ["If th
{"intent": "faq_14", "utterances" : ["Can I opt to defer my medical treatment to a later date for personal reasons, and if I do, will I be penalised?"], "respo
{"intent": "faq_15", "utterances" : ["Can I submit my claim via email (with scanned copies of supporting documents)?"], "responses" : ["Yes, we accept claim su
{"intent": "faq_16", "utterances" : ["Will you still consider accepting my claim, even if I submit it after the 30-day deadline specified in your policy?"], "r
{"intent": "faq_17", "utterances" : ["What are the supporting documents required for travel claim submission?"], "responses" : ["The supporting documents requi
{"intent": "greet", "utterances" : ["Hello", "Hi", "Good morning", "Good evening"], "responses" : ["Hi! What can I help you with today?", "Hi! How may I help y
{"intent": "goodbye", "utterances" : ["Bye", "Exit"], "responses" : ["Have a great day", "Bye!", "Goodbye", "Good day :)"]},
{"intent": "affirm", "utterances" : ["Yes", "y", "correct"], "responses" : ["Is there anything else I can help you with?", "Do you have any other queries?", ]}
{"intent": "deny", "utterances" : ["No", "n", "not really", "incorrect"], "responses" : ["Have a great day", "Bye!", "Goodbye", "Good day :)"]},
{"intent": "request_policy_status", "utterances" : ["Can I get the status of policy number P112343255245?", "What is the status of my policy with number P69456
{"intent": "request_claim_status", "utterances" : ["I want the status of claim number C568894123514", "What is the status of my claim?", "Kindly fetch the stat
{"intent": "request_by_date", "utterances" : ["Can I get the policies that have loss date greater than 25/03/2022?", "I would like the list of policies with lo
{"intent": "request_by_location", "utterances" : ["I want policies from a particular location", "Show me all the policies from US", "Display all policies from
{"intent": "request_by_payment_amt", "utterances" : ["Please fetch the list of claims that have payment amount greater than 6.81.", "Find the list of claims ha
```

FIGURE 2.5: Training Data for SetFit

### 2.4.2 Entity Extraction

We had to extract 5 types of entities:

- Date: To extract dates, we used the python `datefinder` module. It extract all sorts of date like strings from a document and turn them into `datetime` objects. The module finds the likely `datetime` strings and then uses `dateutil` to convert to the `datetime` object. After obtaining the `datetime` object, we converted them to a date string for ease of querying.
- Policy/Claim number: We fixed a policy and claim number format since the policy and claim numbers in the database were all tokenized. The format we chose was the letter 'P'/'C' followed by 12 digits for policy/claim numbers respectively. To extract these numbers, we used a simple regular expression.
- Amount: We used a regular expression for decimal numbers to extract amount.
- Location: We used a subset of all the locations present in the database and searched for those locations in the utterances while ignoring case.

### 2.4.3 Conversational Flow

To handle conversational flow, we have defined several functions that can be invoked in case of happy paths (such as the user providing all the required entities directly, or on the first prompt) as well unhappy paths (such as the user providing bogus input, or the user changing their mind about making the query). The invocation of these functions depends on the values returned by the entity extraction functions. We also put all the string constant in a separate dictionary and used it to improve modularity of our code. The following flowchart gives an overview of the flow of control in our chatbot:

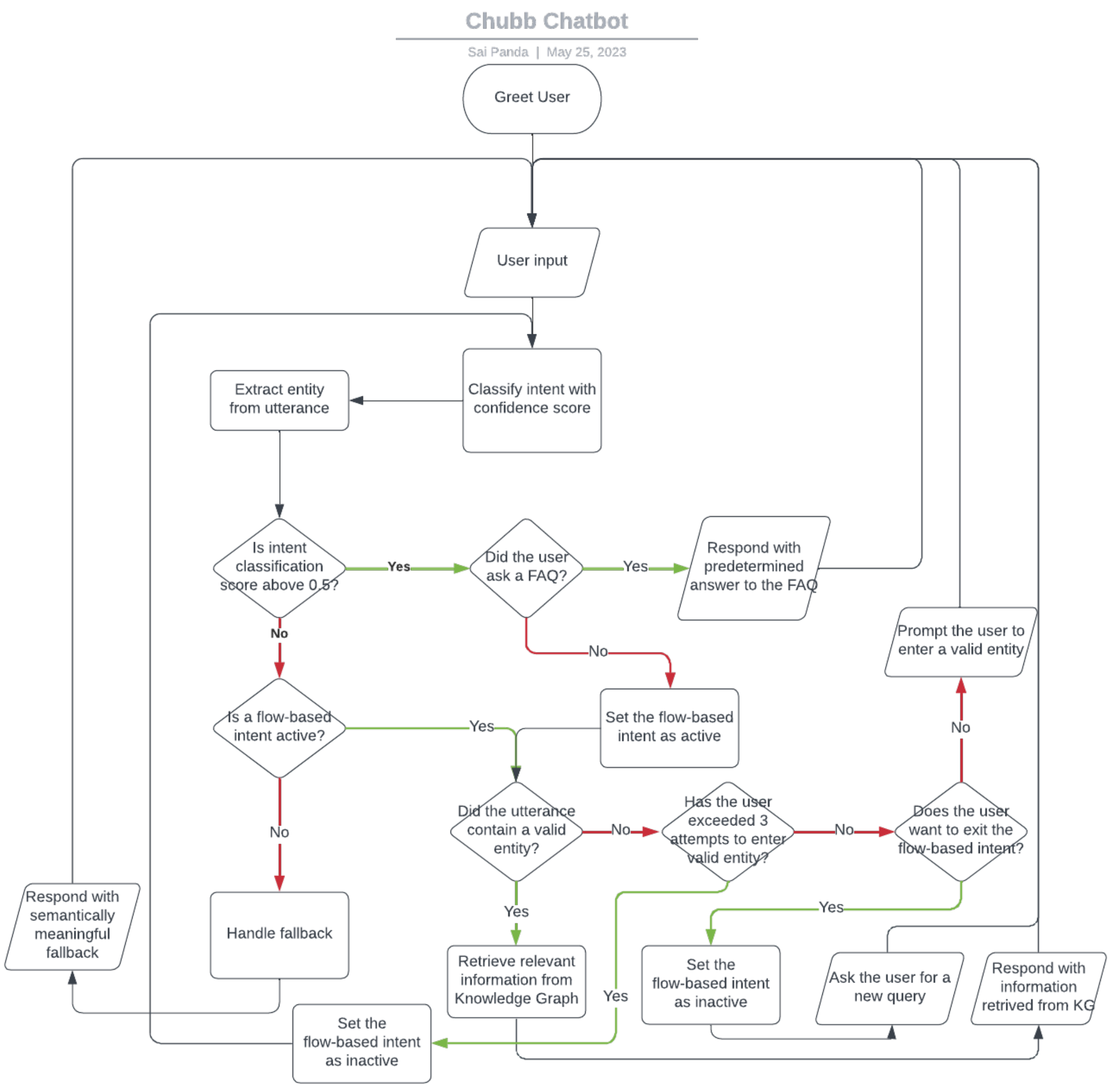

Chubb Chatbot
Sai Panda | May 25, 2023
Greet User
User input
Classify intent with confidence score
Extract entity from utterance
Is intent classification score above 0.5?
Yes
Did the user ask a FAQ?
Yes
Respond with predetermined answer to the FAQ
No
No
Set the flow-based intent as active
Is a flow-based intent active?
Yes
No
Handle fallback
Respond with semantically meaningful fallback
Did the utterance contain a valid entity?
No
Has the user exceeded 3 attempts to enter valid entity?
No
Does the user want to exit the flow-based intent?
No
Prompt the user to enter a valid entity
Yes
Yes
Yes
Retrieve relevant information from Knowledge Graph
Set the flow-based intent as inactive
Ask the user for a new query
Respond with information retrived from KG
Set the flow-based intent as inactive

FIGURE 2.6: Flowchart detailing Conversation Flow

We kept track of active flow-based intents and gave the user three ways to exit them:

- By providing a valid entity: In this case, our chatbot fetches the relevant information from the knowledge graph and displays it to the user.
- By explicitly expressing desire to exit: In this case, we prompt the user for confirmation and if they reply with an affirmation, we exit the flow-based intent.

- By providing an invalid entity 3 times: In this case the chatbot automatically exits the flow-based intent and processes the user input as a fresh utterance, going to fallback handling if required.

### 2.4.4 Fallback

We shift the process the utterance using our fallback responses if the confidence score generated by our SetFit model is below the threshold of 0.5. Finding the appropriate fallback response was a symmetric semantic search task, that is we needed to find most sematically relevant pairs from our fallback responses and the utterance which were of similar length. Sentence embedding models are currently best suited for this task. We decided to use the `all-MiniLM-L6-v2` sbert model for ranking the fallback responses. We generated the embeddings for each of our fallback responses and the utterance and used their cosine similarity scores to find the top 3 most relevant responses and returned them to the user.

### 2.4.5 Gradio

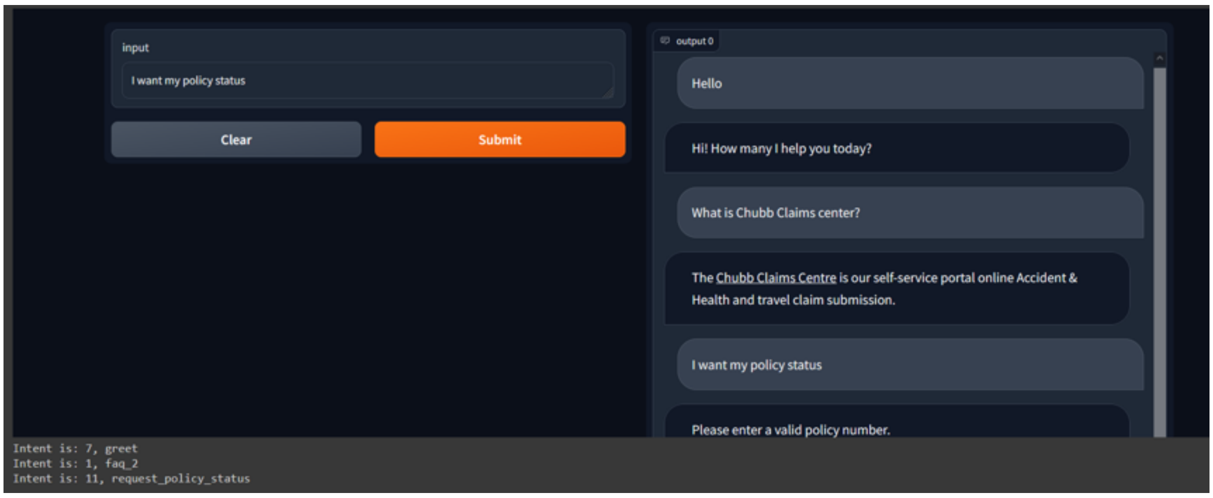
Gradio is an open-source Python library that is used to build machine learning and data science demos and web applications. It is useful for:[1]

- Demoing machine learning models for clients/collaborators/users/students.
- Deploying models quickly with automatic shareable links and getting feedback on model performance.
- Debugging models interactively during development using built-in manipulation and interpretation tools.

FIGURE 2.7: Gradio: Build Machine Learning Web Apps — in Python

Since the command line is not very user friendly, we used Gradio to create a GUI for our chatbot. We modified the `Interface` class to retain history of our messages so that we could go through them by scrolling up.

### 2.4.6 Screenshots



I want my policy status

Hello

Hi! How many I help you today?

What is Chubb Claims center?

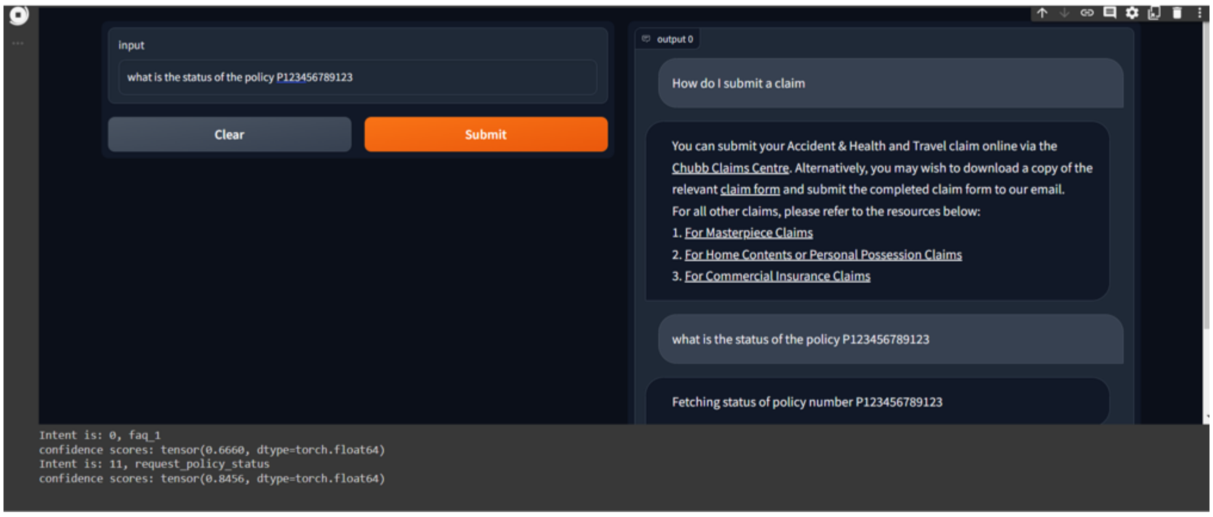
The Chubb Claims Centre is our self-service portal online Accident & Health and travel claim submission.

I want my policy status

Please enter a valid policy number.

```
Intent is: 7, greet
Intent is: 1, faq_2
Intent is: 11, request_policy_status
```

FIGURE 2.8: A screenshot of our current chatbot depicting intent classification using SetFit and regex-based entity extraction, along with conversational flow



what is the status of the policy P123456789123

How do I submit a claim

You can submit your Accident & Health and Travel claim online via the Chubb Claims Centre. Alternatively, you may wish to download a copy of the relevant claim form and submit the completed claim form to our email.
For all other claims, please refer to the resources below:
1. For Masterpiece Claims
2. For Home Contents or Personal Possession Claims
3. For Commercial Insurance Claims

what is the status of the policy P123456789123

Fetching status of policy number P123456789123

```
Intent is: 0, faq_1
confidence scores: tensor(0.6660, dtype=torch.float64)
Intent is: 11, request_policy_status
confidence scores: tensor(0.8456, dtype=torch.float64)
```

FIGURE 2.9: A screenshot depicting the confidence score of the predicted intent in our current chatbot

# Chapter 3

# Knowledge Graphs

## 3.1 What are Knowledge Graphs?

1. A knowledge graph, also called a semantic network, represents a graphical representation of a network of objects, events, situations, and defines the relations between them.

2. A typical Knowledge graph contains typically three main components:

    - **Nodes:** They represent entities such as people, places, or concepts
    - **Edges:** They represent the relationships between these entities.
    - **Labels:** Each node and label is labeled with attributes to help contextualize the information and make it more accessible to users.

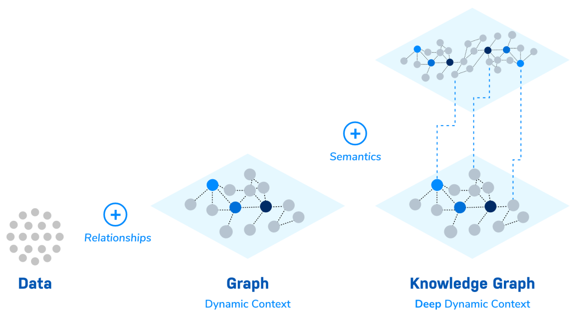


FIGURE 3.1: How data is used for creating a KG

3. It can be created manually or automatically, using techniques like NLP, ML and semantic web technologies. Google uses Knowledge graphs to enhance search results by providing rich, structured information about entities.

4. Knowledge graphs are typically made up of datasets from various sources, which frequently differ in structure. Schemas, identities and context work together to provide structure to diverse data. Schemas provide the framework for the knowledge graph, identities classify the underlying nodes appropriately, and the context determines the setting in which that knowledge exists. These components help distinguish words with multiple meanings.

5. The main benefit of using a Knowledge graph is its ability to connect and organize information in a way that makes it more meaningful and useful. A graph data model makes it easier to explore, analyze and understand large interconnected datasets.

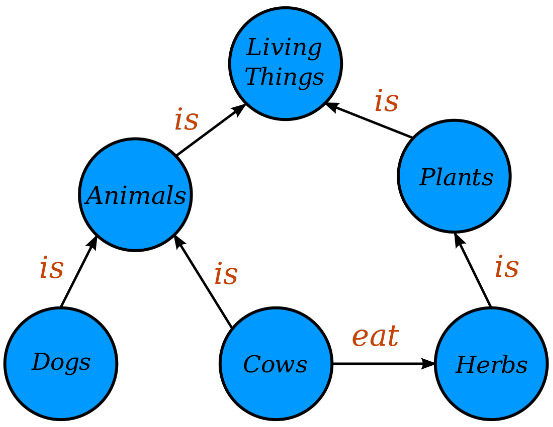


FIGURE 3.2: A simple knowledge graph establishing relationships between animals

Here, **Edges** (Relationships) are : is, eat
**Nodes** are: Dogs, Cows, Herbs, Plants, Living Things, Animals

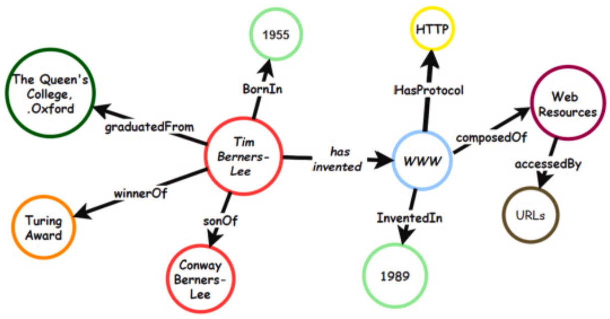


FIGURE 3.3: Another simple knowledge graph depicting relations

## 3.2 How to create a Knowledge Graph?

We use a graph database management system, **Neo4j**, to store, manage and query data.

### 3.2.1 Neo4j

Neo4j uses a graph-based model to store, manage and query data from its created graph database.[2]

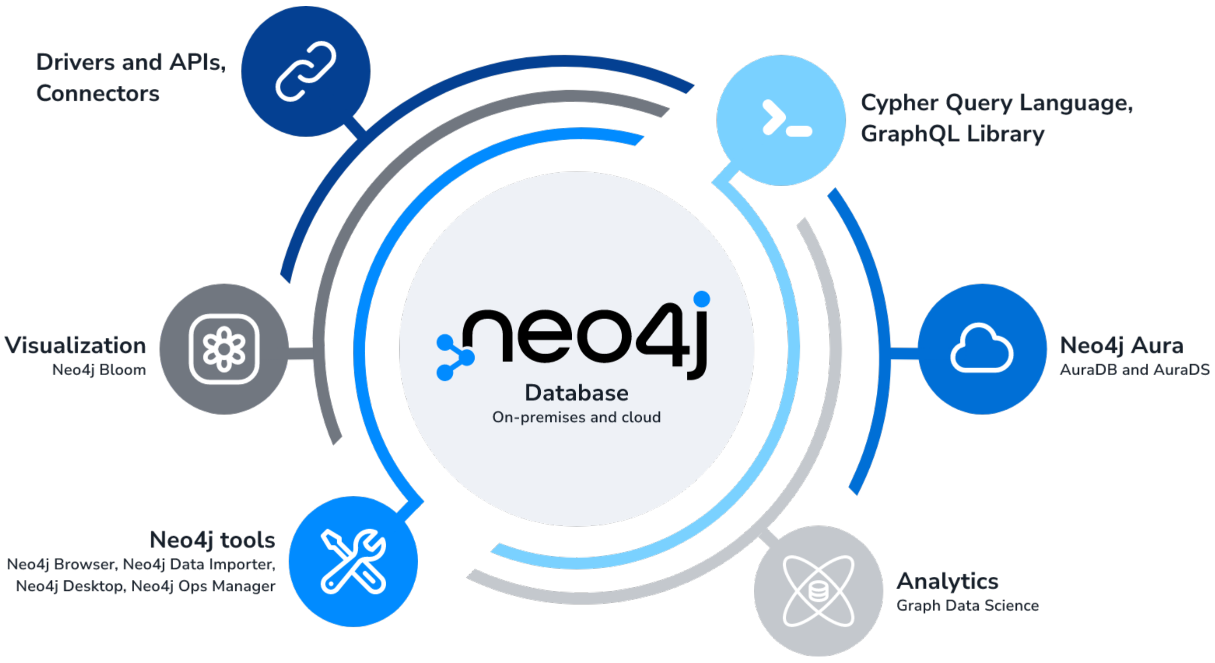

Drivers and APIs, Connectors
Cypher Query Language, GraphQL Library
Visualization
Neo4j Bloom
neo4j
Database
On-premises and cloud
Neo4j Aura
AuraDB and AuraDS
Neo4j tools
Neo4j Browser, Neo4j Data Importer, Neo4j Desktop, Neo4j Ops Manager
Analytics
Graph Data Science

FIGURE 3.4: Features offered by neo4j

1. Neo4j is a graph database management system and allows users to store, manage, and query data using a graph-based model.

2. Users can create and manage these nodes and edges using a query language called Cypher. Cypher is similar to SQL in syntax, but is designed specifically for working with graph data.

3. Applications involve complex, interconnected data sets, such as social networks, recommendation engines, and fraud detection systems.

## 3.3 Creating a Database in Neo4j

1. Create a new database in Neo4j: Open Neo4j Desktop and create a new project. Then create a new database in the project and start it.

2. Import the CSV dataset into Neo4j: Navigate to the "Import" tab in the Neo4j Desktop and select the "CSV" option. Select the CSV file that contains your dataset and configure the import settings (e.g., specify the delimiter, headers, etc.).[3]

3. Design the data model: Before importing the data, it is important to design the data model that will represent your knowledge graph. This involves identifying the entities, relationships, and properties in your dataset and defining them in a graph schema.
4. Define the schema in Cypher: Use the Cypher query language to define the schema of your graph. For example, you might create nodes for entities, relationships to connect them, and properties to describe them.
5. Import the data: Once the schema is defined, use the Cypher query language to import the data from the CSV file into the graph.

## 3.4 Importing Relational Data into Neo4j

Dataset format: `CSV` files
`LOAD CSV` (Cypher command): This command is a great starting point and handles small to medium-sized datasets (up to 10 million records)

```
LOAD CSV WITH HEADERS FROM 'file:///yourfile.csv' AS row
CREATE (n:Entity {name: row.entity_name})
CREATE (m:Entity {name: row.related_entity})
CREATE (n)-[:RELATIONSHIP_TYPE]->(m)
```

- Supports loading / ingesting CSV data from a URL
- Directly maps input data into complex graph/domain structure
- Handles data conversion
- Creates or merges entities, relationships, and structure

The following snippets of code have been used to implement our backend graph database and for creating relationships between nodes in the database.

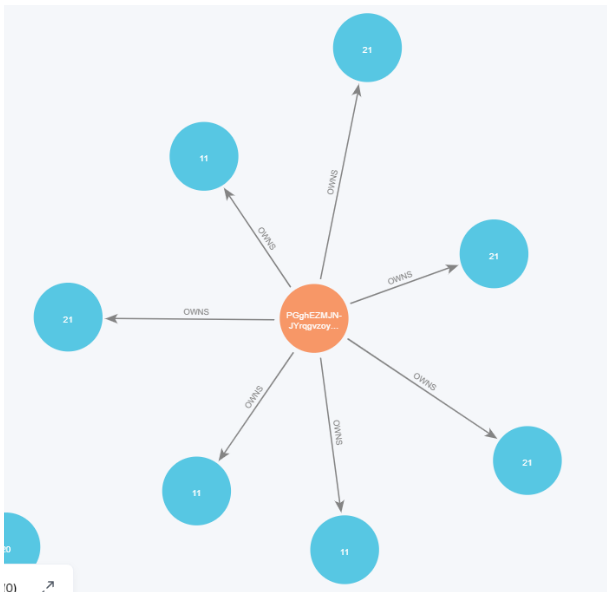


FIGURE 3.5: A limited and specific view of the central (unique party identifier) node connected to the type of policy that the party owns with Chubb with the labels identifying the relationship between the two nodes. The central node has a hash key that uniquely identifies the party that owns policies with the company. The numbers in the blue nodes show the specific type of policy owned.

```
UNWIND $nodeRecords AS nodeRecord
WITH *
WHERE NOT nodeRecord.`Policy_DB_Identifier` IN $idsToSkip AND NOT nodeRecord.`Policy_DB_Ident
MERGE (n: `Policy` { `Policy_DB_Identifier`: nodeRecord.`Policy_DB_Identifier` })
SET n.`Policy_Number` = nodeRecord.`Policy_Number`
SET n.`Policy_Start_Date` = nodeRecord.`Policy_Start_Date`
SET n.`Policy_End_Date` = nodeRecord.`Policy_End_Date`
SET n.`Policy_Type` = toFloat(trim(nodeRecord.`Policy_Type`))
SET n.`Broker_Code` = nodeRecord.`Broker_Code`;
```

FIGURE 3.6: Implementing backend graph database - Part 1

```
CREATE CONSTRAINT `imp_uniq_Policy_Policy_DB_Identifier` IF NOT EXISTS
FOR (n: `Policy`)
REQUIRE (n.`Policy_DB_Identifier`) IS UNIQUE;
```

FIGURE 3.7: Implementing backend graph database - Part 2

```
UNWIND $relRecords AS relRecord
MATCH (source: `Party` { `Party_DB_Identifier`: relRecord.`Party_DB_Identifier` })
MATCH (target: `Policy` { `Policy_DB_Identifier`: relRecord.`Policy_DB_Identifier` })
MERGE (source)-[r: `OWNS`]→(target);
```

FIGURE 3.8: Creating relationships between nodes in the database

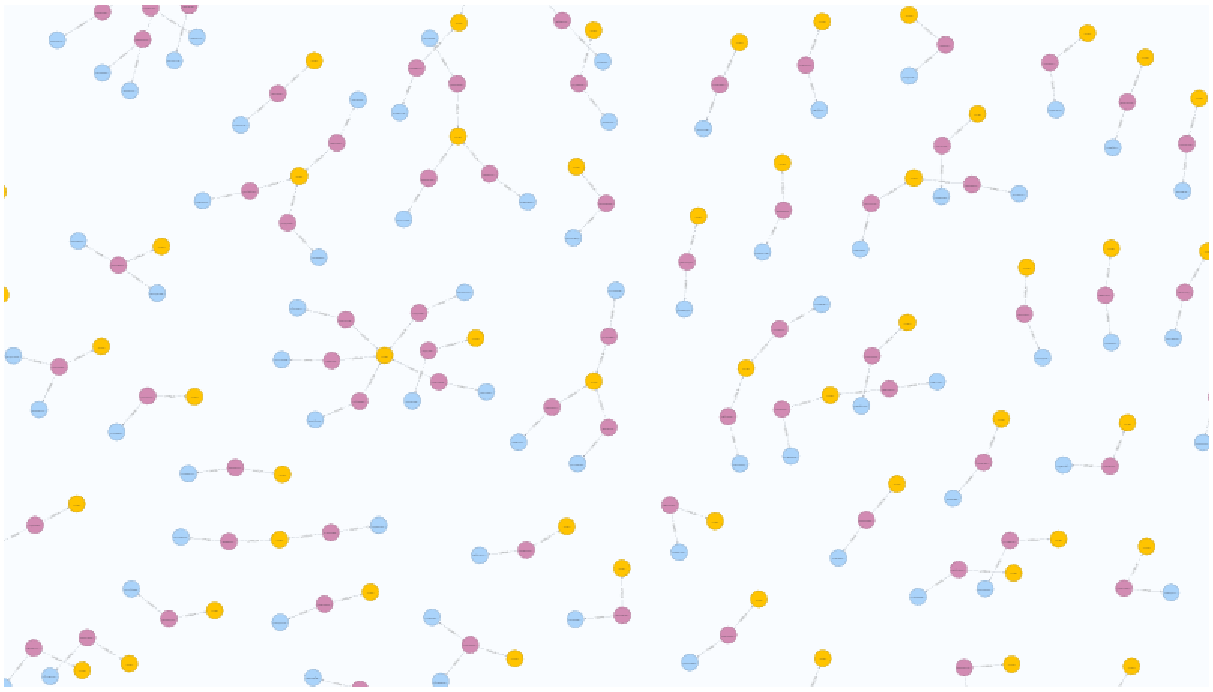


FIGURE 3.9: A macroscopic zoomed out view of our graph database. Nodes are defined as the unique database identifiers of the insurance claimant party connected to many different nodes signifying the location, the policy they own and the location of the claim made against the policy.

## 3.5 What we have done

In the initial stages, our team chalked out a bunch of set methods to set up a database for our chatbot. After extensive research, we realized that knowledge graph implementation of a database as the backend of our chatbot was the most efficient and workable way to go for us.

Using neo4j as defined in the passages above, we set out to import our Chubb data (csv format) into our neo4j sandbox so as to be able to implement a knowledge graph with predefined entities and relationships between those entities. The entities represented by nodes then have directional relationships defined between them so as to allow the chatbot to infer relationships between different named entities. This will allow our chatbot to infer context, and be able to answer more conversational queries rather than just fully typed out questions from a question bank.

Neo4j graph databased are navigational in nature and are not index based. This allows our chatbot to have more flexibility and ability to answer questions in a much more natural and understandable way. We imported our data using CypherQL queries from CSV files with headers. We created unique constraints on all nodes, and used the merge function to create relationships between them so as to have links like `MADE_AGAINST` between Claim and Policy. Further, we added properties for nodes and relationships so we could specify dates, locations and so on.

When it came to hosting our database, we hosted it locally initially and then to facilitate remote and concurrent access for all of us, we shifted to a cloud instance of a Neo4j AuraDB which connects to our chatbot's frontend via a Neo4j database driver and a simple URI connection link. The chatbot can then send queries using query languages to facilitate easy navigation of the linked nodes in our knowledge graph.

# Chapter 4

# Integration

## 4.1 Creating a Cloud Instance

To start with, we create a cloud instance of a knowledge graph on the neo4j console.Each instance is created with a unique instance id, a neo4j username,password(which are required to access the database) and a connection URI.

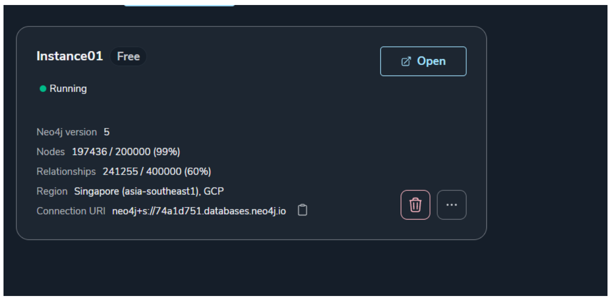
Neo4j supports many connection URI schemes : neo4j:

- neo4j:// (unencrypted) and neo4j+s:// (encrypted with TLS) - work on either a single instance or a cluster. The routing is handled by the driver.
- bolt://(unencrypted) and bolt+s:// (encrypted with TLS) - connect only to the server with the IP you specify. It does not route anywhere else.

Once the cloud instance is created, the knowledge graph is created using cypher queries, i.e. by creating proper nodes and defining suitable relationships between them. This has been discussed above in greater detail. After the knowledge graph is created and populated with relevant data, it can be queried from the deep learning model to extract relevant information.

## 4.2 Querying the Knowledge Graph

To query, we connect to the knowledge graph using the GraphDatabase driver from neo4j using the neo4j username, password and the connection URI. Once the connection has been established and verified, execute manually written cypher queries for each of the intents described. A mapping has been defined between these intents and manually cypher queries instead of converting the user utterance into cypher query language due to the complexity of the task. Finally a local session is


Instance01 Free

● Running

Neo4j version 5

Nodes 197436 / 200000 (99%)

Relationships 241255 / 400000 (60%)

Region Singapore (asia-southeast1), GCP

Connection URI neo4j+s://74a1d751.databases.neo4j.io

FIGURE 4.1: A screenshot of our cloud instance

created and all of these queries are executed using the `session.execute_read()` method. Using this method, we can extract the relevant properties from the knowledge graph.

After this integration, the user can :

- Query for the status of a policy based on the policy number, which gives the status of the claim associated with the policy
- Query for the status of a claim by entering the claim number
- Get all policies (policy numbers) from a specified location
- Get all policies whose loss date is greater than a user entered date
- Get all claims with an amount greater than a specified amount

# Bibliography

[1] *Gradio: Build Machine Learning Web Apps — in Python*. en. URL: https://github.com/gradio-app/gradio (visited on 04/05/2023).

[2] *Graph Data Modeling*. en. URL: https://neo4j.com/developer/data-modeling/ (visited on 04/05/2023).

[3] *Importing CSV Data into Neo4j*. en. URL: https://neo4j.com/developer/guide-import-csv/ (visited on 04/05/2023).

[4] Ajitesh Kumar. *NLP pre-trained models explained with examples*. Sept. 2021. URL: https://vitalflux.com/nlp-pre-trained-models-explained-with-examples/.

[5] *Rasa DIET Blog*. en. URL: https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture/ (visited on 04/05/2023).

[6] *Rasa Glossary*. en. URL: https://rasa.com/docs/rasa/glossary/ (visited on 04/05/2023).

[7] *SetFit: Efficient Few-Shot Learning Without Prompts*. en. URL: https://huggingface.co/blog/setfit (visited on 04/05/2023).

[8] *The Essential Chatbot Terminology for Beginners*. en. URL: https://www.chatbot.com/blog/chatbot-terminology/ (visited on 04/05/2023).

[9] Ashish Vaswani et al. *Attention is all you need*. Dec. 2017. URL: https://doi.org/10.48550/arXiv.1706.03762.

[10] Vladimir Vlasov, Johannes E. M. Mosig, and Alan Nichol. *Dialogue Transformers*. 2020. arXiv: 1910.00486 [cs.CL].

[11] *What is a chatbot?* en. URL: https://www.ibm.com/topics/chatbots (visited on 04/05/2023).

[12] *What is conversational AI?* en. URL: https://www.ibm.com/topics/conversational-ai (visited on 04/05/2023).