14328030 - Octavia Istocescu

# Chatbot Report

## Index Page



## 1. Introduction

The system documented below (“The Weatherbot”) is a chatbot that answers weather-related questions.

As an international student that travels frequently between cities, I am naturally interested in the current and forecasted weather and thought of this coursework as a good opportunity to invest some time in my interest.

In addition, there’s no general weather service to offer a chatbot (that I know of), so I was curious about the implementation challenges and usability of such a system.

## 2. Background

Besides the course and lab material, I only took a little inspiration from a couple of blog posts on chatbots, in order to learn how to tackle the problem, for example:

1. https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e [1]

2. https://towardsdatascience.com/how-to-build-a-basic-chatbot-from-scratch-f63a2ccf5262 [2]

They gave me the general idea of structuring the question analysis around intents and helped with the structure of the intents file.

Differences between these projects and “The Weatherbot”:
- None of them is weather-related;
- They basically just match questions and offer a predefined answer (no information retrieval);
- I don’t use TensorFlow.

# 3. Proposed system

## a. Functionality

“The Weatherbot” is a console-based program that asks the user for a weather-related question and answers it, in an infinite loop.

The system identifies greetings and requests for small talk, but it’s very cursory about answering them, as I wanted to focus my time and energy on the information retrieval side of the project.

Each question contains at least one date and one place (city). The default date is today, the default city is the last one that was explicitly specified (or, if none, the system asks for one). Valid dates are from today to up to seven days into the future, as that’s the limit of the weather service.

The system can do simple queries and comparisons. For example, the following questions should all provide meaningful answers:
- What’s the weather like in London?
- Will it rain tomorrow in Nottingham or Leicester?
- Is it colder in Rome, Naples or Florence?
- Will it be colder in Paris or London on Monday?
- Is it windier in Amsterdam than in London?

## b. Originality

The processing of intents was inspired by the blog posts on chatbots. Also, the NLP technique is based on the lab material, but besides that, “The Weatherbot” is an original work, designed and coded from scratch.

## c. Implementation

The program queries the OpenWeather API (https://openweathermap.org/). The free plan provides the current weather and a 7-day forecast.

The code is structured in 4 python files:

- *weatherbot.py* - The application entry point. It initializes the system and does the main loop that reads user input and responds with an answer. It takes an optional parameter, “-t”, that starts the program in training mode.
- *city.py* - The City class. It stores the city static data (name, coordinates) and does weather queries.
- *query.py* - The Query class. It stores the user query data (question text, tokenized text, filtered text and the intent tag) and does the NLP (question matching and extraction of place and time).
- *intent_functions.py* - One function per intent. They basically construct the answer for a given Query.

There are 2 additional data files:

- *city.list.json* - A list of cities supported by OpenWeather with their geographical coordinates.
- *intents.json* - A list of intents.

An intent is a dictionary with two mandatory keys:

- *tag* - The intent handle. In the current implementation, there are two types of supported intents:
    - general intents (*hi*, *bye*, *chatter* and *debug*); these are processed in the main loop directly (in *weatherbot.py*).
    - weather-related intents (*general_query*, *temp_query*, *temp_compare*, *rain_query*, *rain_compare*, *wind_query*, *wind_compare*); these are called by introspection, so extending this list is as easy as adding a new function in *intent_functions.py* and a new tag in *intents.json*.
- *patterns* - A list of strings representing previously asked questions.

When running in training mode, the program matches each question to a *pattern* and answers the question if the match is perfect. Otherwise, it asks the user if the interpretation is correct, and if it’s not, it asks for the correct interpretation (the correct *tag*). At the end of the loop, the

program saves the updated *intents.json* file. In normal mode, the program just answers the question based on the best match it can find.

The question matching is done by using the cosine similarity of the TF-IDF weighted vectors (bags of words). The manhattan distance and log frequency weighting were also tried but showed poorer results, so they were abandoned.

The program does no stemming or lemmatisation, given that the questions are rather short and the system needs to distinguish between simple queries and comparisons. For example, the following questions should produce different results:

1. *Is it cold in London or in Paris?* (simple query)
2. *Is it colder in London or in Paris?* (comparison)

As the only difference is *cold* vs *colder*, this means that the words should be treated as different tokens for different endings.

## Improvement opportunities

The identification of dates and places from questions is rule-based and not very robust. The question is pre-processed by looking for date-related words and city names, which are extracted (removed) from the question. Only then is the question matched to a *pattern* in the intents database. One problem with this implementation is that we can't have a city name that matches a common vocabulary word. Did you know that Is and Hot are city names? I only found out with this occasion!

After implementing it this way, I came to suspect that using a placeholder in the patterns could provide better results. For example, the question: *Is it cold in London tomorrow?*, in the current implementation, is reduced to the pattern *is it cold in* (after removing *London* as the place and *tomorrow* as the date). An alternate implementation could be reducing it to the pattern *is it cold in <CITY> <DATE>*. A different matching algorithm should be used as well.

Other minor improvements that could provide for a better user experience:

- Spellchecking (or training with enough data that common typos could get into the database);
- Recognizing a wider array of date formats;
- Offering the answer in a few more formats;
- Adding data from the past;
- Adding more intents (queries for pressure, humidity, cloudiness etc.)

# 4. Evaluation

For the system evaluation, I did a manual benchmark of 20 questions:

- 1 point for Actual answer identical to Expected answer;
- 0.5 points for usable Actual answer (information is incomplete, unexpected order of answers, ambiguous interpretation of the question etc.)'
- 0 points for unusable Actual answer.

| # | Question | Expected answer | Actual answer | Pts |
|---|---|---|---|---|
| 1 | Is it cold in London? | <temperature in London> | <temperature in London> | 1 |
| 2 | Is it colder in New York? | <temperature in New York today><br><temperature in London today><br><temperature comparison between New York and London today> | <temperature in New York><br><temperature in London><br><temperature comparison between New York and London today> | 1 |
| 3 | Will it be colder tomorrow? | <temperature in New York tomorrow><br><temperature in London tomorrow><br><temperature comparison between New York and London tomorrow> | <temperature in London tomorrow><br><temperature in New York tomorrow><br><temperature comparison between London and New York tomorrow> | 0.5 |
| 4 | Which one is rainier? | <rain in New York today><br><rain in London today><br><rain comparison between New York and London today> | <rain in New York tomorrow><br><rain in London tomorrow><br><rain comparison between New York and London tomorrow> | 0.5 |
| 5 | Which one is rainier today? | <rain in New York today><br><rain in London today><br><rain comparison between New York and London today> | <rain in London today><br><rain in New York today><br><rain comparison between London and New York today> | 0.5 |
| 6 | Is Paris rainier? | <rain in Paris today><br><rain in New York today><br><rain in London today><br><rain comparison between Paris, New York and London today> | <rain in Paris today><br><rain in New York today><br><rain comparison between Paris and New York today> | 0.5 |
| 7 | Is Paris rainier than New York or London? | <rain in Paris today><br><rain in New York today><br><rain in London today><br><rain comparison between Paris, New York and London today> | <rain in Paris today><br><rain in New York today><br><rain in London today><br><rain comparison between Paris, New York and London today> | 1 |
| 8 | Is Paris rainier today than London tomorrow? | <rain in Paris today><br><rain in London tomorrow><br><rain comparison between Paris today and London tomorrow> | <rain in Paris today><br><rain in London tomorrow><br><rain comparison between Paris today and London tomorrow> | 1 |

| | | | | |
|---|---|---|---|---|
| 9 | What's the weather like in Dubai? | <weather description for Dubai today> | <weather description for Dubai tomorrow> | 0.5 |
| 10 | What's the weather like in Dubai today? | <weather description for Dubai today> | <weather description for Dubai today> | 1 |
| 11 | Will it rain? | <rain in Dubai today> | <rain in Dubai today> | 1 |
| 12 | What about Tokyo? | <rain in Tokyo today> | <temperature in Tokyo today> | 0 |
| 13 | Is it rainy in Tokyo? | <rain in Tokyo today> | <temperature in Tokyo today><br><temperature in Tokyo today><br><temperature comparison between Tokyo and Tokyo today> | 0 |
| 14 | Is it windy in Tokyo? | <wind speed in Tokyo today> | <wind speed in Tokyo today> | 1 |
| 15 | Is it rainier in London? | <rain in London today><br><rain in Tokyo today><br><rain comparison between London and Tokyo today> | <rain in London today><br><rain in Tokyo today><br><rain comparison between London and Tokyo today> | 1 |
| 16 | Is it rainier in Paris? | <rain in Paris today><br><rain in London today><br><rain in Tokyo today><br><rain comparison between Paris, London and Tokyo today> | <rain in Paris today><br><rain in Tokyo today><br><rain comparison between Paris and Tokyo today> | 0.5 |
| 17 | Which one is rainier: Paris, London or Tokyo? | <rain in Paris today><br><rain in London today><br><rain in Tokyo today><br><rain comparison between Paris, London and Tokyo today> | <rain in Paris today><br><rain in Tokyo today><br><rain in London today><br><rain comparison between Paris, London and Tokyo today> | 0.5 |
| 18 | I want to know if it rains on Monday. | <rain in Paris on Monday><br><rain in London on Monday><br><rain in Tokyo on Monday> | <rain in London on Monday> | 0.5 |
| 19 | Ok nice talking to you! | <small talk response> | <rain in Nice on Monday> | 0 |
| 20 | Bye! | <program termination> | <program termination> | 1 |
| | | | **TOTAL** | **13** |

# 5. Discussion

## a. Results

For the Evaluation phase, the system got a 13/20 score (65%). Let's discuss the main points of failure:

- The context for dates needs improvement (Q4, Q9) - better disambiguation between requests for current weather vs forecast.
- The context for cities needs improvement (Q6, Q16, Q18) - better discrimination of which cities should be included in the query.
- There is no context for the query type (Q12) - the program doesn't keep track of *what* was asked, only of *where* and *when*.
- More training data is needed (Q13, though a very simple question, failed because "rainy" is not in the training vocabulary and the rest of the words are ambiguous enough; Q19 failed because the small talk data is super sparse).

## b. Impact

With the addition of voice recognition, the system could be integrated into weather applications and provide a quicker way of investigating the weather on the go.

## c. Fairness and bias

The system is biased towards the larger ("more important") cities. Currently, there is no way to specify exactly which city the user is interested in if there are more than one with the same name (the larger one will be auto-selected). If the user asks about London, it's implicitly assumed it's London, UK, not London, France, London, Canada or London, Arkansas, USA.

# 6. Conclusion

While the system is functional, it needs testing with more than N=1 users, and more labelled data is necessary.

# Reference List


1. P. Pandey, 17 Sept 2018, "*Building a Simple Chatbot from Scratch in Python (using NLTK)*" , <https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e>
2. P. Shivaprasad, 20 Aug 2020, "*How to Build a Basic Chatbot from Scratch*", <https://towardsdatascience.com/how-to-build-a-basic-chatbot-from-scratch-f63a2ccf5262 >