

---

## Interim Report

By: Xiaofeng Fu

Supervisor: Anthony Hunter

Used project title: London Trip Information Advisor

Current project title: London Restaurant Advisor

During the implementation of the project, I have changed the project title since the previous one, London Trip Information Advisor, involves many fields and it is difficult for me to implement all of them before the deadline of the project. Therefore, now I only focus on providing advises of London restaurants in my project.

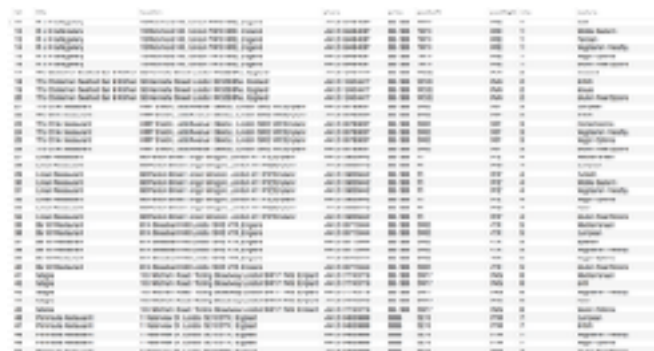
### Progress:

In the November project plan, I set several requirements in the MoSCoW table:

- A valid source of London trip information
- The ability to identify and understand users' questions
- The ability to find the potential answer from the source.
- The ability to translate the potential answer to an answer written in natural language.
- A website user interface to contain this system.
- Put the website on a server.
- A beautiful website user interface.

Until now, most of the requirements are implemented:

- To get sufficient restaurants data, I write a web scrawler to collect more than 18,000 sets of London restaurants information in a mySQL database from the website of TripAdvisor.

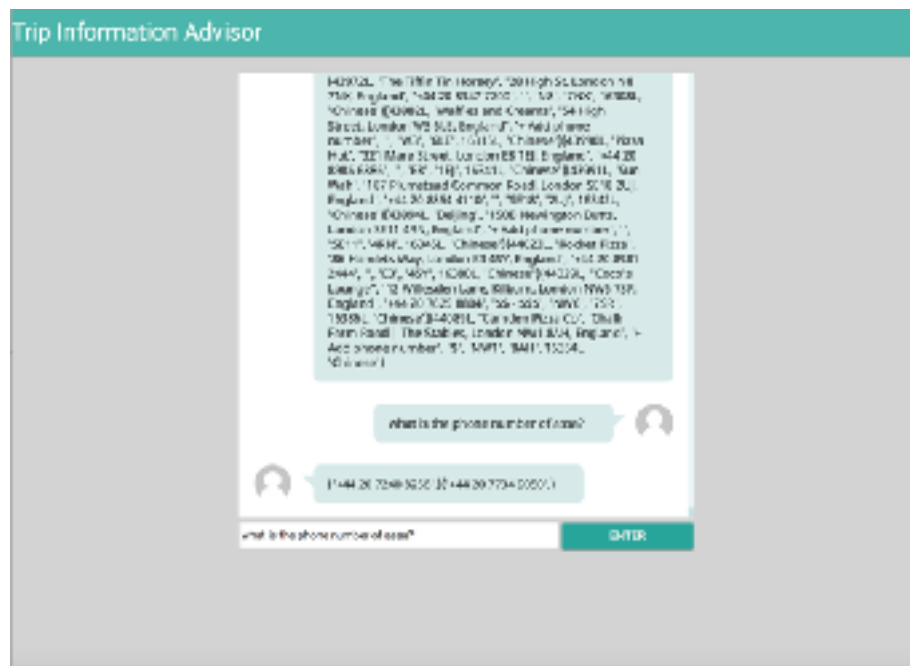


The image shows a screenshot of a MySQL database table containing restaurant information. The table has multiple columns, including restaurant names, addresses, phone numbers, and other details. The data is organized in a grid format, with rows representing individual restaurant entries. The text is somewhat small and blurry, but the structure of the table is clearly visible.

- I write a parser to analyse users' questions and find the potential answer from the database. This parser includes four parts. The first part is query formulation. For this step, the system will extract all potential keywords from the query. Firstly, the system will ignore all stopwords and punctuations in the query. Secondly, entities will be tagged based on the database. Then the system will use POS-tagger in NLTK to tag every word in noun phrases to separate all adjective and adverb from the noun. The second part is answer type classification. Because I am focusing on a closed domain QA system, question types are fixed and it is not difficult to estimate what questions will be asked by users. The parser uses Jaccard Coefficient mechanism to implement this part. This mechanism is using a equation:  $\text{probability} = \frac{(\text{Interaction of QueryTerms and user question})}{(\text{Union of QueryTerms and user questions})}$  Firstly, I wrote a survey to collect potential questions from my friends. Then, I built a database to save the type of question and the related answer type in a special form. And then, I built a function to compare each word of user question with each query terms to find the number of word interaction. The query term with the highest probability is the type of the user question. For example, 'what is the best Burger King in London', the query term 'what is the adjective entity in loc' will be matched and the answer type will be 'description or definition'. Then it will be easier for me to determine the SQL query. The third part is relation extraction. The parser bind each keyword with its description in a tuple, ('Burger King', 'Entity'), using a external dictionary and the restaurant database. The forth part is SQL query generation. Based on the

known answer type and the set of keywords, the parser will form a SQL query and get the corresponding data from the database.

- For the user interface part, I use Django to build this website chat window. This chat window includes a simple chat feature which allow users to ask questions and get answer from the system.



### Remaining work:

- For natural language generation part, I have implemented a simple NLG system to generate many single sentences. The problem of this system is that it cannot recognise other grammar rules and it is not flexible. Therefore, I am working on a new NLG system.
- Another work is to put the database on a cloud server.