Nutri-Assist: Voice-enabled dialogue assistant for getting information on food nutritions

MD Sami Uddin, Ibrokhim Iskandarov
muddi004, iiska001 [@odu.edu]
Department of Computer Science, Old Dominion University

I. Changes from Mid

Discarded the whole system to check food items from the local database(CSV in our case). Instead, now we are using RESTApi to call FDC fooddata endpoints and getting the nutrition values out of JSON response. Also, before mid, for each food item(extracted from user command), we were calculating the cosine similarity of all the food-names in the local food-database. Now that we are getting items specific to the user uttered food item using API, things are more accurate when we're applying cosine similarity with them and ranking them according to that value. Moreover, the code is more organized and modular than before. Comments are added too! Link to Github: https://github.com/rayansami/nutrition-assistant

***Abstract*—Currently there are two types of the system in the market right now, one is a search interface, where users place food names by typing and get some results and select among the list manually. And another one is generic assistant like Google assistant/Siri, which redirects the user to the search engine. Hence there is no system that stays on the same interface and gives the consumer the desired nutrition values and lessen the manual intervention. Hence, this work targets to fill this gap and merge functionality between the personal assistants and manual searchable applications. In short, our system will take a casual voice input, search for the appropriate food items and their nutritional values and show them to the user in an effortless way. This will free the users from boring scrolling and select items , and prevent distraction from changing up the interface(Assistant to search engine). Besides, Nutri-Assist will help the consumers to improve their experience by remembering the precious choices and improving upon the time.**

II. Introduction

In modern days the influence of Natural Language Processing(NLP) has become very important in human lives. This branch of artificial intelligence got such a broad spectrum of applications because the algorithms aiding computers to understand people as we do are becoming more powerful and efficient. Despite many breakthroughs in the technology of NLP, it is not a simple task to teach machines the way how we communicate. Some of the common factors which may hurt the performance of a given NLP can be but not limited to local speech slang, background noise, how similar the sequence of words is, etc.

Since one of the main roles of using NLP is to simplify to some extent the interaction of humans with computers, we will develop a system that will assist users in gaining information about their requested food nutrition. Also, our system will allow users to assert their interests casually. After making an extensive search on the internet, we concluded that there are not many applications that may perform the above-specified tasks. The overall expected format of the program will be to prompt the user to speak out the food nutrition he wants, and output all relevant information such as calories, energy, nutrition elements, etc. One of the main resources to obtain information about the specified food nutrition will be a 3rd party API provided by the United States Department of Agriculture(https://fdc.nal.usda.gov).

The report is organized as follows: in section 2, we will describe the importance of diet management applications in human lives. Since we are developing a program that will directly interact with the user, we will also discuss previous research works devoted to the solution of similar problems. Additionally, in this section, we will emphasize a little bit more about information retrieval. This section will also cover basic techniques when it comes to a complex dialogue between the user and the machine. In later parts, we'll go through the technical details of the system as well as our methods for studying the usefulness.

III. Literature Review

Before starting our work we went through other works that are close to our project. We divided this into four sub-components and briefly discussed our findings on them.

A. State Of Acceptance To Diet Management Applications

That's being said, you are what you eat!

Nowadays, the world is facing an unprecedented level of obesity and numerous diseases such as cancer, diabetes, and other chronic illness because of unhealthy food habits. Just in the USA, the percentage of obese people covers 35% of adults and 17% of teenagers.[1]

However, diet planning is becoming increasingly popular among people in general. Especially with the help of smartphones and the internet, it's easier than ever to track food intake and count the calories. Even the older adults are more leaning towards getting virtual assistance [2]. And there have been some applications that focus on helping its

consumers by guiding them in managing weights, eating better and effectively.

Studies have shown that people are more likely to use apps if they perceive them as effective and useful[3]. And there are quite a several popular mobile applications out in the market. To name a few, there is an app named Calorie Counter Food Diary(https://www.mynetdiary.com/). Users can set nutrition and weight goals and can scan barcodes while shopping which helps to make wise decisions. It also lets users track different nutritional values. Another app is Food Intolerances App(https://www.baliza.de/en/apps/histamine.html). Not all food is suitable for everybody and this varies from person to person. This app helps people with allergies such as histamine intolerance, mastocytosis, fructose malabsorption, sorbitol intolerance, gluten sensitivity, and lactose intolerance in choosing the right one.

So it's evident that our system will be highly welcome if it serves the purpose effectively and effortlessly and lets users have what they want.

B. Works Towards Designing Dialogue Systems

We are going to develop our system where we take voice input and show results after inferring the command. That's why, In this section, we will briefly discuss the other applications of NLP which found their niche in solving real-world problems. The main idea behind all of the products we mention below is based on simple receiving input in terms of speech from the user and respond with the required action based on customer needs. Despite the novelty of NLP technology, it already existed since the early 1990th. Back at that time, the functionality of early dialogue assistants was limited to telephone-based call routing systems [4], weather information systems [5], travel planning [6] etc.

More recent developments of the dialogue systems include more advanced tasks such as car navigation, entertainment, and communications [7]. These systems are also called a multi-domain which involves several different tasks all at once. Some examples are telemetries, smart home, and intelligent robots. These systems have gradually become capable of supporting multiple tasks and accessing information from a broad variety of sources and services.

C. Information Retrieval

Information retrieval is an activity and science that works towards finding relevant information from a collection of resources. These resources can be text, images, audio, videos, or metadata that represents data and searches can be full-text or other content-based indexing.

IR is also a query-based process where the user enters something and the system provides related information. However, the difference between IR and classical SQL queries of a database is, the results from the IR system are typically ranked and it is often expected that they do not match with the user's query. It provides a computed numeric score for each result that matches the user's query and ranks according to that. Users get to see the top-ranked result and the query can be refined if the user wishes[8]

We plan to take voice input to our system, convert it into text format and mine the information about food names from that text.

Often it's a case that users have a mental model for their queries, but they lack the knowledge of how an IR system works. Whether the query is short or long, the effectiveness varies. Often the query needs to be further clarified. It's also important to understand which query needs to be clarified. To solve this problem, the authors presented a technique that can determine and can avoid unnecessary queries and maintain performance [9]. From our project's perspective, this will be a very useful technique. Our system will have to determine if it needs further information when a user says "I ate beef" or "I had 1/2 lb beef brisket". When a user says "I ate beef", the IR system of our software will decide if it needs to ask the user about the quantity. Furthermore, a user can ask "Calories in my soup", in which case the IR system will detect that there are many types of soups in the market and it will ask further questions to have specific information about the soup from the user.

D. Partially Observable Markov Decision Processes For Spoken Dialogue Systems

In this paper, the authors are discussing possible issues when dealing with a Spoken Dialogue System (SDS). It becomes particularly useful when user commands are constrained into digits, place-names, and some short commands. But unfortunately, it becomes error-prone when dealing with complex speech domains. In other terms, once the task complexity increases, the accuracy of SDS decreases rapidly.

To improve SDS performance, there are three methods: confidence scores, automated planning, and parallel dialogue hypothesis each of these methods are used depending on the type of error we are dealing with. The main problem occurs once we combine these approaches to resolve all types of complexities. This is caused by the lack of an overall statistical framework.

One solution that the authors are proposing is a partially observable Markov decision process (POMDP) that can enable a proper framework [10]. We will see that POMDP is a more general approach and all three methods mentioned above are just special cases of POMDP.
POMDP can be defined as a tuple [11],
{S, A, T, R, O, Z, k, b0}

- S is a set of states
- A is a set of actions
- T is a transition probability from one state to another given the action

- R is expected real values reward
- O is a set of observations
- Z is observation probability
- k is a geometric discount factor
- b0 is an initial belief

The paper describes the definition of each of these arguments and how they are related. The final goal will be to calculate at each time-step, the belief state distribution b. One of the important points in this article covers not only the transition to the regular next state solely depending on the current state but also discusses the situations when the next state may depend not only on the current state but also on few last states. This paper integrates the application of POMDP to all three approaches in resolving the errors in detail.

Even so, despite the clear potential of POMDPs, several key challenges remain. Most crucially, scaling the model to handle real-world problems remains a significant challenge: the complexity of a POMDP grows with the number of user goals, and optimization quickly becomes intractable.

In the current section, we have discussed in some detail the principle of POMDP. We mentioned how important this concept could be when dealing with dialogue systems. We also discussed that the POMDP is a complex structure that unifies all existing methods in identifying various types of errors. Additionally, despite the fact of promising applications of this method, there still some open questions related to this topic.

The relevance of this topic to our work is the following, to outline for the user the proper food nutrition, we may also maintain the program to prompt a few questions (sometimes consisting of more) the user.

Here is a simple illustration,

- User: ”Cheese”
- Our System: specify which type of cheese
- User: ”cheddar cheese”

Next time, when the user wants cheese, the system will show the result for cheddar cheese as a first result.

IV. Technical Overview

Fig 1 shows the bird’s eye view of the architectural approach of the nutrition assistance system. The user can turn on the microphone and speak casually to interact with the assistant. When the assistant gets the natural language voice command, it starts the procedure of analysis and exploration.

After receiving the spoken command, the nutrition assistant first converts the voice into text using Automatic Speech Recognizer (ASR). With the whole text on hand, the interaction manager interprets the covert intent and decides if more information is needed from the user. It also uses previously interpreted results and past responses to make the analysis, and formulate a new response based on the context. If the analysis comes out with the decision that there’s a shortage of enough information, it passes the control to Dialog Generator which then asks the users to be more specific about their choices. And this loop further iterates until the interaction manager decides upon the specificity of the input command.

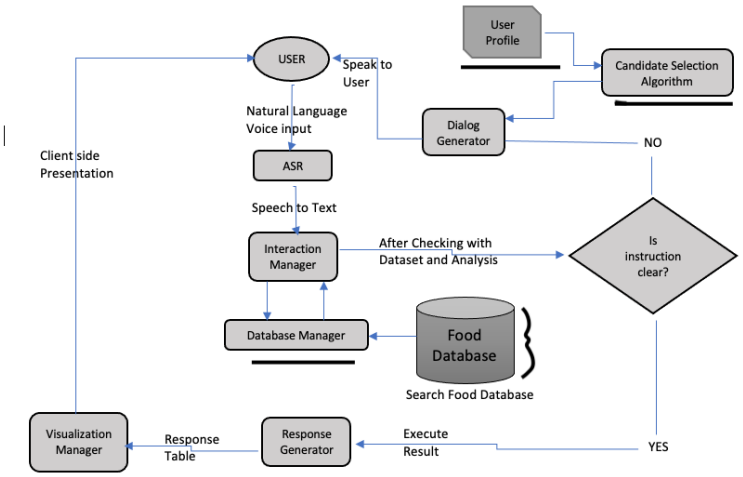

Fig. 1. Nutri-Assist architecture and workflow diagram

After determining the correct context, the data is passed to Task Manager which searches for the food items from a data storage. It also matches the quantitative amount mentioned in the input with the one found in food data storage and calculates the nutrition accordingly(apply unitary method if necessary). After collecting the necessary data, the task manager passes them to the Response Generator, which arranges everything in tabular format so that the Visualization Manager can present them to the user in an understandable manner.

A. Automatic Speech Recognizer

After using the microphone and receiving the voice, we are using Google ASR web service[9] for converting speech to text. It is publicly available and free to use. Then we pass the text to our Interaction Manager to proceed further. It takes only a couple of seconds to get the text response from the speech.

B. Interaction Manager

This Manager is in charge of a couple of things and the heaviest part of this project. Initially, it takes on the text that was converted from the natural language voice command. This module separates the food items and then runs an analysis to check if any more information is needed for any of the items listed. So the procedural sequence of this block can be divided into the following three sub-components: the first one is isolating Food Elements, the second would be Interacting with Database Manager, and the other is Analysing Command Lucidity.

1) Isolating Food Elements: The primary step on having a deep understanding[12] and processing digital speech in a dialog system is the ability to label the speech with the functional tags. Our Nutri-Assist must be able to get the contextual definition of the speech by leveraging tagging or annotation. In our case, when the user says ”Half pound brisket with mashed potato and a cup of coffee for lunch”, Nutri-Assist will separate half-pound brisket, mashed potatoes, and a cup of coffee. As a conjunction ”with” is used, Nutri-Assist will also look for food items that have brisket and mashed potatoes together. These will be handled by Task Manager anyway. The sole responsibility of this sub-component is to split the items so that these can be used later in the Task Manager.

2) Interacting with Database Manager: For every separated food item, the system will call the Database Manager and pass the list of items. This manager holds the food storage database and manages all the interactivity between that database and the Interaction Manager.

Our food database consists of the same food with different combinations. So It is necessary to find out if the user is seeking some kind of combo dish or just an item. Hence to find out the proper match, the Database Manager runs each candidate on the list through the database. Then it assigns a score to each of the prospective matches based on cosine similarity and picks the one that has the highest score. After that, it sends back the findings to the core module of the Interaction manager.

We are currently using the FDC food database and it contains food item names against a unique ID. When the Task Manager decides upon a food item from the storage, it reads the FDC-id and searches for the FDC web interface. After getting a resulting page, Task Manager scrapes the resulting nutrition values and the measurement mentioned in the page and sends those data to the Response Generator to proceed further.

3) Analysing Command Lucidity: After having the isolated food items and operation associated with the Database Manager, this part checks if the list of items is clear enough for going into the next steps. If the interaction with the user is fairly new and there’s yet to build any state based on his/her choices, then the Interaction Manager will invoke the Response Generator with the current findings from the food database and persist the choices in the memory.

On the other hand, if there’s some state about the user’s previous choices already, then the Manager will do the analysis based on that, and decide if it does need to ask further questions. Let’s imagine a scenario where the isolated food item from the user’s voice command is ”Soup”. If the user already chose ”Tomato Soup” previously, then Nutri-Assist will go ahead and finish the rest of the process based on tomato soup. But if there’s nothing like this already, the system will decide to ask the user to be more specific about the choice.

C. Dialog Generator

The task of the Dialog Generator is to ask the user to be more specific about the choice(s). This module is invoked by the Interaction Manager if Nutri-Assist decides that the voice commands are insufficient to proceed further. When it is only (”Fried Chicken”) then the Interaction Manager may not decide to invoke the Dialog Manager if it finds something close to this already. But if it’s (”Soup”) and there are many alternatives with this choice, then the Dialog Manager will be invoked and will speak out to the User to be more specific about the choice of the soup.

1) Candidate Selection Algorithm: The Dialog Manager decides on asking more specific questions if there are too many options found in the database for that particular choice. After retrieving the choice, this module will rank them from most matched food item to least matched food item. Then it will ask the user about the top three findings where the user gets to say yes or no. If the user says yes to any of those options, the program will go to the next phase with that choice. And if the user says no then the system will restart and ask the user to give a more specific command

2) User Profile: When these three options are asked by the system and one of them is selected by the user, the system will store which option was chosen for what command. That way, next time when the user provides the same command, the system will know which food item the user implies.

D. Response Generator

This is mainly the front-side of the back-end of Nutri-Assist. After getting the available data from the scrapped webpages, it will gather them in JSON or XML format and send the dictionary of the gathered data to the Visualization Manager.

E. Visualization Manager

The Visualization Manager is responsible for the representation of the extracted data to the users. It listens to the Response Generator and when receives anything from that, it arranges the data in the client-side system. In our consideration, the client-side front-end can be anything from the web page to the app interface. It is also responsible for easing the experience of the user to see the important nutritional element and their corresponding values.

F. Illustration: Putting It All together

Let’s imagine a scenario where the user says ”I had peanut butter and a cup of coffee for breakfast” after opening up the system interface. To serve the speech, Nutri-Assist will carry out the following sets of operation:

1) Convert casually spoken command of the user into text
2) Determine the food items. Here the extracted items would be ”peanut butter” and ”coffee”.
3) Extract the given measurements of the food item and establish the connection between them. Here, the system finds ”a cup” is related to the ”coffee”.
4) Analyze the candidates and decide whether the instructions are clear or more information is needed.

- If instructions are not clear narrate to the user to be more specific about the choices

5) Extracted candidates are processed against a dataset containing food names.
6) For each item prepare a tabular representation for the user on the client-side.

V. User Study with Nutri-Assist

To find out the usefulness and effectiveness, we'll conduct a study with users to evaluate the acceptance of our system. We'll aim at two different hypotheses as follows:
Hypothesis 1: More efficient than regular multi-functional personal assistants like Siri, Alexa, or Google Assistant.
Hypothesis 2: More usable than the applications out there which only allows users to find their desired item via manual typing, scrolling, and selecting.

A. System Study

Currently, we are studying the functionality and possibilities in ourselves before moving into the real users. The task is to use voice command and get the proper nutrition values for desired food items.

We have divided the type of user command into several cases:

1) User asked only one item that is directly matched with the food database
2) User asked only one item that is directly matched with the food database, but there are many options for the same match, which in a sense creates a vague command
3) User asks multiple food items and each of them matches directly with the item in the database
4) User asks multiple food items but one of them(or each of them) has so many matches in the database. Which also creates vague commands.

Also, We divided the user scenario into two parts:

1) If the user uses the system for the first time
2) If it's the returning user and giving the same or different command
3) If the desired item is not found using the system

We shall look into the responsiveness and accuracy of the system. For the first time user, it's straightforward and gives just the results. For the returning users, there can be a case where he/she has a history of choice, which we need to go over before showing the result. We also need to monitor the correctness and adaptability of the system as a whole.

VI. Acknowledgements

This project was developed for CS 795/895: TPCS of NLP course. The progress and development were closely monitored and mentored by the course instructor Dr. Ashok. Other students from the class helped us find the lack of our development and helped us get the proper design pipeline. The content is the sole responsibility of the authors.

VII. Conclusion and Future Work

As future work, we are planning to implement POMDP as well as maintain the user profile to give a personalized experience. Also, we will develop a web interface so that the transactions feel smooth and hassle-free.

References

[1] N. B. Johnson, L. D. Hayes, K. Brown, E. C. Hoo, and K. A. Ethier, "Cdc national health report: leading causes of morbidity and mortality and associated behavioral risk and protective factors—united states, 2005–2013," 2014.
[2] T. A. Festervand, D. B. Meinert, and S. J. Vitell, "Older adults attitudes toward and adoption of personal computers and computer-based lifestyle assistance," *Journal of Applied Business Research (JABR)*, vol. 10, no. 2, pp. 13–22, 1994.
[3] B. Okumus, A. Bilgihan, and A. B. Ozturk, "Factors affecting the acceptance of smartphone diet applications," *Journal of Hospitality Marketing & Management*, vol. 25, no. 6, pp. 726–747, 2016.
[4] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may i help you?," *Speech communication*, vol. 23, no. 1-2, pp. 113–127, 1997.
[5] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington, "Juplter: a telephone-based conversational interface for weather information," *IEEE Transactions on speech and audio processing*, vol. 8, no. 1, pp. 85–96, 2000.
[6] M. A. Walker, A. I. Rudnicky, J. Aberdeen, E. O. Bratt, J. S. Garofolo, H. Hastie, A. N. Le, B. Pellom, A. Potamianos, R. Passonneau, *et al.*, "Darpa communicator evaluation: Progress from 2000 to 2001," in *Seventh International Conference on Spoken Language Processing*, 2002.
[7] W. Minker, U. Haiber, P. Heisterkamp, and S. Scheible, "The seneca spoken language dialogue system," *Speech Communication*, vol. 43, no. 1-2, pp. 89–102, 2004.
[8] W. B. Frankes and R. Baeza-Yates, "Information retrieval: Data structure & algorithms," *PrenticeHall, Englewood cliffs, NJ*, 1992.
[9] G. Kumaran and J. Allan, "Adapting information retrieval systems to user queries," *Information Processing & Management*, vol. 44, no. 6, pp. 1838–1862, 2008.
[10] J. D. Williams and S. Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.
[11] S. Young, M. Gašić, B. Thomson, and J. D. Williams, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.
[12] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer, "Dialogue act modeling for automatic tagging and recognition of conversational speech," *Computational linguistics*, vol. 26, no. 3, pp. 339–373, 2000.