INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 6, Issue 1)
Available online at: www.ijariit.com

Linux based Virtual Assistant in C

Ritwik Sharma
ritwik.1204@gmail.com
University of Petroleum and Energy Studies, Dehradun, Uttarakhand

Riya
riyad17031998@gmail.com
University of Petroleum and Energy Studies, Dehradun, Uttarakhand

ABSTRACT

In this paper, we propose a C-based approach to develop a Virtual Assistant for the Linux operating system. The goal is to perform tasks with ease so that the user effort is minimum. We are giving input sentence in the form of Natural Language which is processed by our Pattern classifier after which virtual assistant is performing the tasks based on the results of pattern classifier. Pattern Classifier's result is the name of a category such as a restaurant, weather, media, etc. based on the highest score among all the categories. We are using pattern classifier, which is much like Multinomial Naive Bayes, the difference is that instead of calculating probabilities this approach is comparing scores of categories which makes this algorithm much simplified. This algorithm is naive because it consists of different independent "features", in this case: words. Each word is considered as a different entity and does not show any relation with other words in the sentence that is being classified.

Keywords— *Virtual assistant, Natural language processing, Artificial intelligence, Pattern classification, Personal digital assistant*

1. INTRODUCTION

Virtual Assistants not only does saves time but also eases the work of the user. It makes our lives stress free, as we do not have to plan our trips without going through weather forecast or we do not have to remember the song list or we can search for nearby restaurants or YouTube search, everything can become so soothing and relaxing [1]. Earlier versions of Virtual Assistants were mostly based on direct matching in which there were limitations such as the problem to store a large amount of data for pattern matching or the problem in understanding different natural languages. Virtual Assistants nowadays works on pattern classifiers which are constructed using some algorithmic approach [2].

Since the 1950s, Naive Bayes has been studied extensively. It remains a popular method for text categorization. It judges the category according to the frequencies of the word. It is based on Bayes probability theorem [3]. It was used for text classification which makes it best suited for our project. It is known for its simplicity and also for its effectiveness. This was the most suitable and balanced algorithm that we analyzed. It was able to choose the best of the provided choices. This algorithm is based on a common principle i.e. each word has its own particular frequency that does not depend on other words frequency. For example, fruit can be considered as orange if it is orange, round, and about 12 cm in diameter. Each of these features contributes to independently classifying orange as a fruit, and there can be no possibility that this feature correlates with some other fruit's features [4]. The main contribution of the paper is the implementation of Simplified naive Bayes algorithm for the development of virtual assistant on Linux based Operation System and the major part is that the whole coding took place in C programming language. Simplified naive Bayes algorithm is easy to implement and use. This method is inspired by the "Multinomial Naive Bayes" [5] algorithm. Implementation of "Multinomial Naive Bayes" algorithm was pretty difficult in the C programming language, which led us to the development of a Simplified version of it.

The contents of the paper are organized as follows. Section A defines the Pattern Classifier i.e. Simplified Naive Bayes approach. Section B defines the approach and algorithm for Google and YouTube Search. Section C defines the approach and algorithm for finding restaurants in an area. Section D defines the approach and algorithm for Weather Forecast. Section E discusses the approach for Playing Multimedia files.

1.1 PATTERN CLASSIFIER

Let {C1, ..., Cn} be the set of sums of each word in a matching Category. {C1, ..., Cn} are then compared with each other. The category which has the maximum C value is selected and the function corresponding to that category is performed.

Let the weather and greeting categories have the following predefined sentences:

Category: weather	**Category: restaurant**
"how is the weather today?"	"Please find some restaurants?"
"show outside condition"	"find places to eat"
"show me the weather forecast?"	"show restaurant in this area?"

Let's classify a few sample input sentences:

input: how is it outside?
term: how (category: weather)
term: is (category: weather)
term: it (no matches)
term: outside (category: weather)

classification: weather (score=3)

input: show some restaurants?
term: show (category: weather (2), restaurant (1))
term: some (category: restaurant (1))
term: restaurant (category: restaurant (2))
classification: restaurant (score=5)

In the First input sentence, weather consists of a maximum number of matching keywords, as weather category scored highest so its functionality will be executed.

In the Second input sentence, the restaurant consists of the maximum number of matching keywords, as restaurant category scored highest so its functionality will be executed.

The main steps are the following:

i. Assign category (different functionalities).
   Example: Weather, Google, media, restaurant, etc.
ii. Assign keywords.
   Example: Greeting - how are, you, etc.
iii. Tokenization
iv. Each token passed to each category through each keyword.
v. If (word==keyword) increment score by 1
vi. Compare: score of categories
vii. Input sentence => category with highest score
viii. Execute => functionality

1.2 GOOGLE SEARCH AND YOUTUBE SEARCH

Google generates a random URL for its each and every search like this:

https://www.google.co.in/search?hl=en&dcr=0&source=hp&ei=_1tOWpGbI5-UvQTuuqKYBw&q=artificial+intelligence&oq=Artifi&gs_l=psy-ab.3.0.35i39k1l2j0j0i131k1j0l6.1242.3817.0.5656.9.7.1.0.0.0.254.990.2-4.5.0....0...1c.1.64.psy-ab..4.5.1014.0..0i67k1.292.xDSaXQ0sCqk

instead of:

https://www.google.co.in/artificial intelligence

The URL was analyzed, and a pattern was noticed in the URL i.e. it replaces the user-search in its URL at a particular place i.e. the text just after **&q=artificial+intelligence**
The URL also made us notice that if the user search consists of more than one word with spaces than space is replaced by the + symbol (see Figure 1).

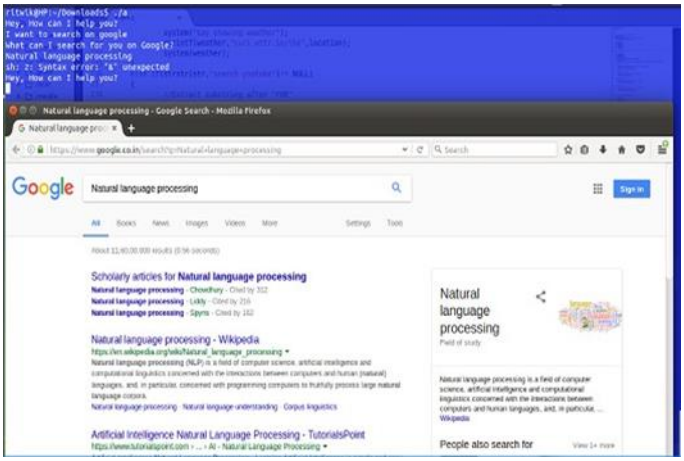

Fig. 1: Google Search for Natural language processing

After analyzing the whole URL, we developed an Algorithm:

i. Input (string)
   Example: String = Artificial Intelligence
   Replace blank spaces with "+".
   Example: String = Artificial+Intelligence
ii. Manipulate Google URL => embed the string to URL
iii. Define buffer to hold the string.
iv. Save URL => buffer using sprintf
v. Use buffer value to execute

The same procedure is followed for YouTube Search.

1.3 FINDING RESTAURANTS IN AN AREA

The approach that makes it different is that the results of the search are displayed on the terminal. For implementing this approach, we used two Googles APIs.

- **Google Geocoding API:** Geocoding is the process of converting addresses (like Taj Mahal, Agra, India) into geographic coordinates (like latitude 27.173891 and longitude 78.042068), which can be used to know the exact location of a place on a map.

- **Google Places API:** The Google Places API is a service that returns information about places defined within this API as establishments, geographic locations, or prominent points of interest using HTTP requests.

Fig. 2: List of restaurants in New York

Google places API takes input for a location as coordinates i.e. (Latitude, Longitude). It does not work with human-readable address due to which we need to find the coordinates of the place in which Google geocoding API helps us. Google Geocoding API converts the location name entered by the user into coordinates.

The coordinates received from the Google geocoding API are passed to Google places API and it processes the coordinates and executes the list of restaurants with name, address, ratings and timings which we later display in the form of tables (see Figure 2).

1.4 WEATHER FORECAST

We are using **http://wttr.in/** to get the weather forecast of a location entered by the user (see Fig. 3).

*http://wttr.in/**location_name***

The location name entered by the user is passed to the above URL in place of the **location name**.

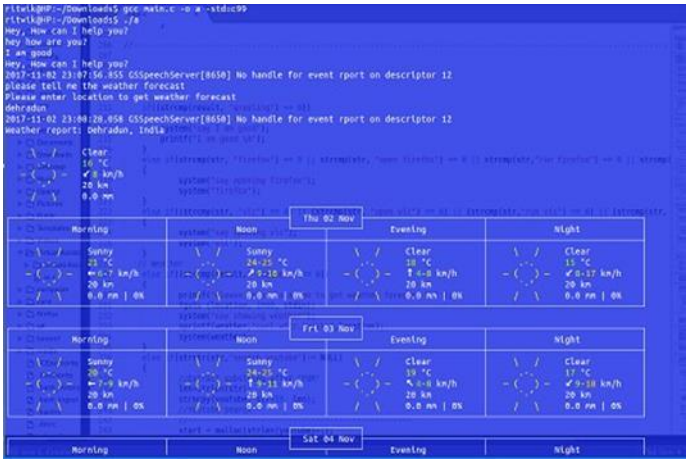

Fig. 3: Weather forecast for Dehradun

1.5 PLAY MULTIMEDIA

Multimedia files i.e. audio and video files are stored in a directory and we use **ls** command to list names of multimedia present in that directory.

The user is asked to enter the multimedia file name out of the list shown. At last, the file name entered by the user is played with the default media player as shown in Figure 4.

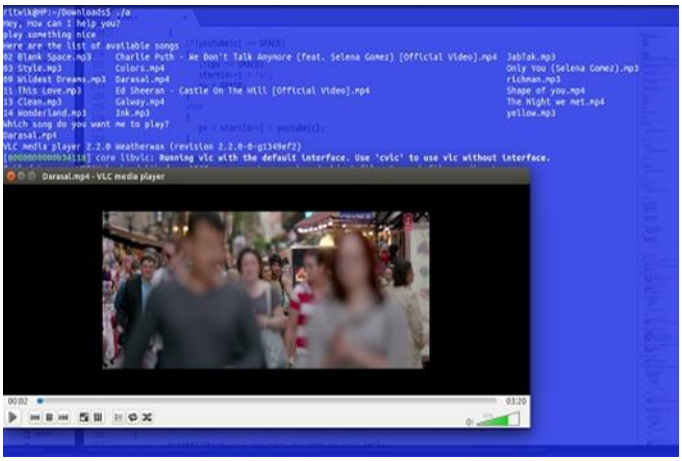

Fig. 4: Playing a video song

2. CONCLUSION

This paper presents the use of Pattern classification and Linux for the development of Virtual Assistant that can understand the user's sentence to some extent at which it can distinguish what functionality the user wants it to perform. It can tell whether the user wants to play a song, find a restaurant, search on Google or YouTube, see the weather forecast, etc. Like, if the user meant weather forecast, it will simply show the weather forecast and asks again **Hey, what can I do for you?** to perform more functionalities.

3. REFERENCES

[1] H. Chung, M. Iorga, J. Voas and S. Lee, "Alexa, Can I Trust You?," in Computer, vol. 50, no. 9, pp. 100-104, 2017. doi: 10.1109/MC.2017.3571053

[2] B. G. Batchelor, N. L. Ford and B. R. Wilkins, "Family of pattern classifiers," in Electronics Letters, vol. 6, no. 12, pp. 368-369, June 11 1970. doi: 10.1049/el:19700258

[3] Russell, Stuart; Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (2nd ed.). Prentice-Hall. ISBN 978-0137903955.

[4] Zhang, Harry. The Optimality of Naive Bayes (PDF). FLAIRS2004 conference.

[5] Rennie, J.; Shih, L.; Teevan, J.; Karger, D. (2003). Tackling the poor assumptions of Naive Bayes classifiers (PDF). ICML.

BIOGRAPHY

Ritwik Sharma
Student
B.Tech
School of Computer Science
UPES, India.

Riya
Student
B.Tech
School of Computer Science
UPES, India.