Chatting Partner for English Language Learners

Kunti Sandanshi¹, Ashwini Bande², Samruddhi Jumale³, Pushkar Shinde⁴
¹,²,³ Department of Computer Engineering, NDMVP KBTCOE, University of Pune, Nashik, India
⁴Assistant Professor, Department of Computer Engineering, NDMVP KBTCOE, University of Pune, Nashik, India

Abstract- This paper discusses the practical implementation of an android application which can be used for entertainment and educational purposes. We discuss a chatbot that will help users to improve their linguistic knowledge of vocabulary and grammar for the English language and also act as a chatting partner to them. The goal is to make users confident to use the language and form complete and correctly spelled sentences. We have provided two modes; first mode, normal chatting with the system without language check. The second mode is Language check; here we check the user’s input statement for grammatical and spelling errors. The output is a detailed description of the mistakes if any. For the implementation, we have used android studio, PhpMyAdmin, and TextGear API. We have also provided Spoken language technology to make language learning effective.

Index terms- Artificial intelligence, Language learning, Natural language processing, Speech recognition.

I. INTRODUCTION

The English language is the most used language in the world. Most of the traditional ways of learning a language include books, in-person coaching classes and online coaching. The drawback of it is the necessity of availability of a tutor, the need to follow certain timings, lack of guarantee of this tutor to be as knowledgeable as he claims, and lack of fun in the process. To learn new things effectively we must be in the mental comfort to process and analyze new information. Also to learn a new language we must practice in that language as much as possible. Forming sentences without any context can make the learner lose interest. Nowadays, Chatbots are used in many applications, one of which is language learning. It is found that users' response to a virtual friend is much more positive when they are trying to learn new things. They converse freely without the fear of social stigma [1]. Learning using a chatbot in one’s time of ease makes learning effective.

To address all these drawbacks of traditional learning and explore the advantages of a chatbot we propose a new system that can act as a chatbot as well as a tutor to the user. The system will be an android application which will provide two modes to the user. First, normal chat, wherein the app will provide output to the user’s query without any grammar check. Second, chat with grammar and spell-check. Thus, a user can use the tutor mode of our app when he/she wants to learn and a chatting partner mode otherwise. The availability of both of these modes makes sure that the app provides both entertainment and educational value to our users.

Most of the chatbots developed on the procedure of recognizing patterns in the input sentence and responding from a template. Another method is a keyword-based matching system. The drawback of this is that it does not take into account the semantics and relationship between the words in the original query [2]. We have used another approach; we have tried to find a similarity between two sentences rather than the pattern. In this approach, we use Levenshtein distance for finding similarity. It finds relatedness between sentence pairs which is an NLP task. It finds if the two sentences are paraphrased or not. We have used a question_answer database which saves a statement and corresponding answer. PhpMyAdmin is used to store the databases on the localhost. It is a freely available tool to handle MySql databases allowing developers to perform complex operations using its UI.

The app is designed to emulate the look and feel of modern-day popular messaging apps such as messenger and WhatsApp. We have also provided the option of speech output. This would help in improving the dialect of the user. For grammar and vocabulary checking we need wide vocabulary and
linguistic rules building which is complex and can decrease accuracy. TextGear API is a freely available linguistic resource. TextGear performs word segmentation, semantic mapping, and syntactic parsing. It would find grammatical and spelling mistakes in the user’s sentence and provide appropriate grammar structure for our users.

II. RELATED WORK

A. A Web-Based Chinese Question Answering Answering Validation.
A web-based Chinese automatic QA system uses an answer extraction method which is based on the calculation of sentence similarity between question and answer [3]. It analyzes the question asked by the user and understands it’s type and corresponding answer type. The validation process involves first choosing the best answer based on the parameter of similarity and if the answer is found with rules, the procedure ends. Further validation is done by choosing a correlation between question and answer.

B. A survey of grammar checkers for natural languages.
Grammar checker generally takes input in the text format from the user and generates the grammatically corrected output or provides suggestions related to it, if the text words are grammatically incorrect[4]. It also replaces the incorrect word with the correct word or suggests the user about the grammatical mistake.

C. A Virtual Context-Adaptive Chatting Partner for Foreign Language Learners
CSIEC is a chatbot that chats with the user with the intention of teaching him/her the language[5]. The user gives an input then the system checks if the input text contains any idiom if it does then response is generated directly following the muster-matching concept.

D. A Rule-Based Style and Grammar Checker
The style and grammar checker detects errors. In this, each word of the input text is assigned a part-of-speech tag and each sentence is split into chunks like nouns and phrases. The text is then matched with all the defined error rules[6]. If the match is true, the text is flagged into the group of errors. The position of the error is noted.

III. METHODOLOGY

The app has a two-tier architecture based on the retrieval-based model. We are using PHP with the XAMPP server and MySQL for manipulating the data at the server-side. The network requests would be between the app and the databases in JSON format. Android Volley is a networking library that provides connectivity to our app. By default all the volley network calls work asynchronously, so we don’t have to worry about using async tasks. We have used in-build Google speech to text and text to speech API for allowing users to communicate with the app in speech format. [Figure 1] represents a complete system architecture.

A. User Scenario
The application provides access to two kinds of user: admin and users.

![Figure 1. System Architecture](image_url)
each user with a unique user_id is saved into the
database.
User_role helps the system differentiate between the
two kinds of users during login.

B. Interaction design
Our application is designed for normal chat(friend
mode) and language checking(grammar and spelling
errors). In the normal chat interaction scenario, the
user’s query would be compared with the questions
in our ‘question answer’ dataset to find similarity.
The similarity is found between the user's query and
statements in the question column of our database.
Levenshtein distance is used for processing natural
language queries to SQL queries to the database.
Levenshtein_ratio() returns the percentage value
indicating the ratio of similarity between the two
given strings. This method increases the accuracy of
the system. It would try to find questions in our
database with levenshtein_ratio >= ‘90%’. If not
found it will look for levenshtein_ratio >= ‘80%’. If
there is no question which satisfies ratio>=80% then
the system will provide a default reply[see Figure 8].
Simultaneously this query is added to
‘not_present_database’. Also, the user can choose to
provide input and receive output in speech format. In
the grammar checking scenario, the user’s input will
be passed to TextGears API. A request is sent along
with input text and access key to the online API. The
response is received from the API in JSON format. It
has the error array which needs to be interpreted at
the client-side(see Figure 7).
Interaction design for the admin panel includes
viewing the questions which are not present in the
dataset. Admin can also add a statement and answer
through the UI(see Figure 9). The database system is
cumbersome for non-expert SQL users, our system
makes sure that people with no knowledge of SQL
can also feed data to the Knowledge Base. The input
can be provided in speech format too. is as The
feeding and retrieving of data takes place in the form
of JSON array.

IV. RESULTS

Table 1. Implementations and Results

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Description</th>
<th>Implementation</th>
<th>App Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>There are two defined roles with which the user can log in. Admin and User</td>
<td>if ($user_role==&quot;1&quot;) // &quot;success_admin&quot;; else if ($user_role==&quot;2&quot;) //echo &quot;success_user&quot;;</td>
<td>Figure 2</td>
</tr>
<tr>
<td>2</td>
<td>User/Admin login</td>
<td>select * from user where credentials like '&lt;username&gt;' and password = '&lt;encrypted password&gt;'</td>
<td>Figure 3</td>
</tr>
<tr>
<td>3</td>
<td>Registration Activity is ONLY available for users</td>
<td>Required field: $name= $_POST['name'] $phone = $_POST['phone'] $password=$_POST['password'] $email = $_POST['email'] $address = $_POST['address'] $gender = $_POST['gender']</td>
<td>Figure 4</td>
</tr>
<tr>
<td>4</td>
<td>Chat Activity</td>
<td>The chat data is displayed on the user screen using RecyclerView</td>
<td>Figure 5</td>
</tr>
<tr>
<td>5</td>
<td>Chat features</td>
<td>a)Chat without grammar correction b)Chat with grammar correction</td>
<td>Figure 6</td>
</tr>
<tr>
<td>6</td>
<td>Grammar detection</td>
<td>curl_setopt($ch, CURLOPT_URL,&quot; <a href="https://api.textgears">https://api.textgears</a> .com/check.php&quot;); curl_setopt($ch, CURLOPT_POST, 1); curl_setopt($ch, CURLOPT_POST_FIELDS, http_build_query(array('text' =&gt;</td>
<td>Figure 7</td>
</tr>
</tbody>
</table>
In this paper, we explored an application that will provide two modes to the user. Practically this approach is complex. The interaction mechanism requires network calls to the databases at every step. The use of a volley library decreases the time complexity of the process up to a certain extent. Studies show that users are not likely to use a learning app for more than an hour in a day. This led us to add the functionality of normal chatting mode. We assume that users want both modes in the application and are likely to use both of them equally. The principle of string similarity does not take into account the relationship between the user’s previous query and the current query. The relationship or pattern between user’s queries i.e conversation flow cannot be recognized by the system as a human does. We need to use deep learning to recognize this pattern.

The gap between the user statement and the one we choose using the Lavenstein function can lead to unrelated responses called generation gaps. The use of a Function instead of artificial network algorithms makes our domain small. The expected variation between user_query and statements in the database must be minimal otherwise the system will generate default reply.

VI. CONCLUSION

Chatbots and conversational agents are applications of NLP. The current prototype aims to provide the user with a friend as well as a tutor. The normal mode would answer according to the user’s input. The tutor mode will provide complete and correctly spelled sentences. The combination of both these functions makes it a promising field. The system can be optimized by reducing interaction costs to improve time efficiency. Improved version of the prototype includes introducing AI algorithms for deep learning and human-computer interaction to make the system much more accurate. As future research lines, first, we try to make the system optimal and time-efficient. Second, we will add the functionality of providing an output which defines various ways to construct the same sentence. Third, we try to make the system feasible to use for blind people and expand it to other languages.

REFERENCES