وزارة التعليم العالي والبحث العلمي
جامعة ديالى
كلية التربية للعلوم الصرفة
قسم الحاسوب

Control of computer by human speech

المشروع مقدم الى قسم الحاسوب في جامعة ديالى
كجزء من متطلبات نيل الحصول على شهادة
البكالوريوس في الحاسوب

للطالب

مجاهد جاسم شكر

بأشراف

الدكتورة سميرة عامر عبدالقادر

٢٠١٩

CHAPTER ONE

INTRODUCTION

1.1 Introduction

Speech is one of the most powerful forms of communication between humans; hence, it is the researchers" ambition in the human computer interaction research field to improve speech interaction between the human and the computer in order to simulate human-human speech interaction. Speech interaction with modern networked computing devices has received increasing interest in the past few years with contributions from Google, Android and IOS. Because they are more natural than graphic-based interfaces, spoken dialogue systems are beginning to form the primary interaction method with a machine [1]. Therefore, speech interaction will play a significant role in humanising machines in the near future [2].

Much research work has focussed on improving recognition rates of the human voice and the technology is now approaching viability for speech based human computer interaction. Speech Interaction splits into more than one area including: speech recognition, speech parsing, NLP (Natural Language Processing), keyword identification, Chabot design/personality, artificial intelligence etc.

Our system is a computer program that have the ability to control computer by human using Natural Language Speech.

Human-Computer Speech is gaining momentum as a technique of computer interaction. There has been a recent upsurge in speech based search engines and assistants such as Siri, Google Chrome and Cortana. Natural Language Processing (NLP) techniques such as NLTK for Python can be applied to analyse speech, and intelligent responses can be found by designing an engine to provide appropriate human like

responses. This type of programming is called a speech analysis, which is the focus of this study.

Artificial intelligence control of computer using speech are written in Python with the nltk package. The Natural Language Toolkit (NLTK) is a platform used for building programs for text analysis. One of the more powerful aspects of the NLTK module is the Part of Speech tagging.

We should use pre-processing stage which include tokenize text using Natural language processing NLTK in python to process voice and insert it into the computer.

Natural language processing is the ability of a computer program to understand human language as it is spoken.

The Natural Language Toolkit (NLTK) is a platform used for building programs for text analysis. The platform was originally released by Steven Bird and Edward Loper in conjunction with a computational linguistics course at the University of Pennsylvania in 2001.

Finally, process of tokenize text using NLTK It includes inserting a complete sentence and then cutting it into words and then performing computer control based on cut words, Our system will reduce time and voltage and helps people with **special needs**, especially those who have lost their hands.

1.2 Research problems

There are problems we should solve using our system like:

1- Our system helps people with **special needs**, especially those who have lost their hand or who have paralysis.

2- **Slow :** Our system will resolve this issue where the response is quick as soon as you speak.

3- The system will reduce **voltage** and **time**.

1.3 Research Objective

The goal of the research is the use of natural language toolkit nltk to create a system that can control the computer using input voice by process voice to text and tokenize text using Natural language processing NLTK in python to process voice and insert it into the computer using artificial intelligence.

١, 4 Research Scope

In this research, a survey of control computer by human speech system design techniques in speech conversation between the human and the computer is presented.

The research focuses on the use of Natural Language Toolkit (NLTK), which is one of the branches of Natural language processing (NLP) for processing input voice and converting them to text, then do the processing of the text entered and cut into words and then do the process of controlling the computer based on these cut words.

CHAPTER TWO

THEORETICAL ASPECT

2.1 Human-Computer Speech interaction

Speech recognition is one of the most natural and sought after techniques in computer and networked device interaction has only recently become possible (last two decades) with the advent of fast computing. Speech is a sophisticated signal and happens at different levels: "semantic, linguistic, articulatory, and acoustic" [3]. Speech is considered as the most natural among the aspects of human communication, owing to copious information implicitly existing beyond the meaning of the spoken words. One of the speech information extraction stages is converting speech to text via Automatic Speech Recognition (ASR) and mining speech information [4]; then, the resulting text can be treated to extract the meaning of the words. Speech recognition is widely accepted as the future of interaction with computers and mobile applications; there is no need to use traditional input devices such as the mouse, keyboard or touch sensitive screen and is especially useful for users who do not have the ability to use these traditional devices [5]. It can help disabled people with paralysis, for example, to interact with modern devices easily by voice only without moving their hands.

2.2 Natural Language Toolkit (NLTK)

In order to deal with and manipulate the text resulting from speech recognition and speech to text conversion, specific toolkits are needed to organise the text into sentences then split them into words, to facilitate semantic and meaning extraction. One of these toolkits is the widely used NLTK which is a free plugin for Python. The Natural Language ToolKit (NLTK) is a set of modules, tutorials and exercises which are open source and cover Natural Language Processing symbolically and statistically. NLTK was developed at the University of Pennsylvania in 2001allowing

computational linguistics with three educational applications in mind: projects, assignments and demonstrations [6] [7]. It can be found within the Python Libraries for Graph manipulation GPL open license. NLTK is used to split words in a string of text and separate the text into parts of speech by tagging word labels according to their positions and functions in the sentence. The resulting tagged words are then processed to extract the meaning and produce a response as speech or action as required. Different grammar rules are used to categorise the tagged words in the text into groups or phrases relating to their neighbours and positions. This type of grouping is called chunking into phrases, such as noun phrases and verb phrases.

2.3 SPEECH ANALYSIS AND RESPONSE

Speech analysis can be divided into three stages: (i) voice recognition and conversion to text, (ii) text processing, and (iii) response and action taking. These stages are explained as follows: Firstly, speaker independent speech passes through a microphone to a digital signal processing package built in the computer to convert it into a stream of pulses that contain speech information. Specific instructions can be used to read input speech then to convert it into text. This stage provides speech text for processing in the next stage. The diagram which illustrates this stage is shown in Fig.2.1.

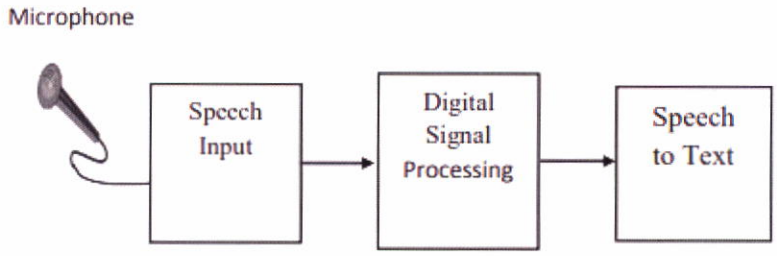

Fig. 2.1. The stage of speech recognition and converting to text

Secondly, the resulting text is split into separate words for tagging with parts-of-speech labels according to their positions and neighbours in the sentence. Different types of grammar can be used in this stage to chunk the individual tagged words in order to form phrases. Keywords can be extracted from these phrases by eliminating unwanted words in chinking operations. These keywords can be checked and corrected if they are not right. The phases of the text processing stage are shown in Fig. 2.2

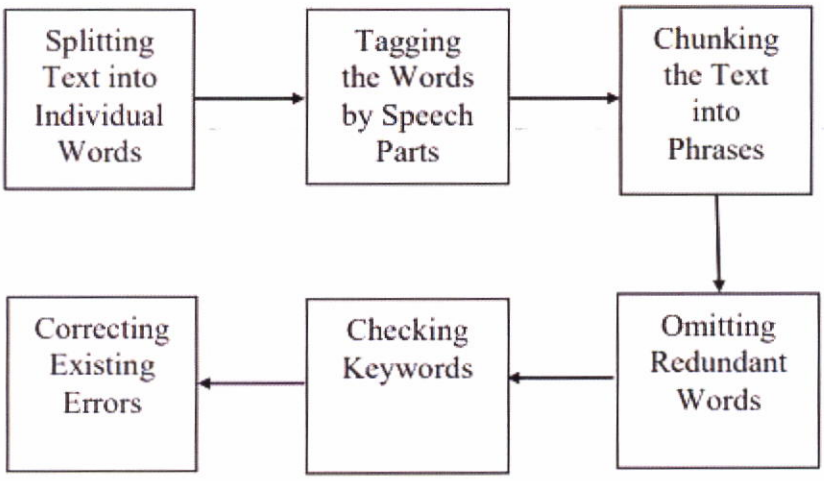

Fig. 2.2 The Stage of Text Processing

Finally, a system can be built to give the desired intelligent response to a natural language speech conversation. The input to this system is keywords released from the speech text processing; the output is the programmed response, which will be, for example, an application running

or any other text or speech response. Fig. 2.3 shows a brief diagram of the the third stage

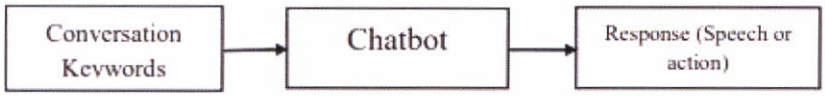

Fig2. 3. The Stage of Response and Action Taking

2.4 Main Parameters of SPEECH ANALYSIS AND RESPONSE

Conversation techniques between a human and a computer can be either chatting by typing text or speech dialogue using the voice. The processing of the information in both techniques is the same after converting speech to text in the case of speech dialogue. A diagram showing the main steps of analysis and processing required to perform human computer conversation is shown in Fig2. 4.

The main parameters which affect human computer interaction quality in conversational systems design are: (i) the techniques used to analyse the text using different grammar sets to produce keywords, (ii) pattern matching techniques used inside the control of computer and depend on a variety of data base access techniques and (iii) the type of response according to the specific application. The focus in this survey is mainly on control of computer system design techniques and a comparison is made between them in terms of the software used, the contribution to the research field in new techniques, and the breadth and depth of the knowledge base used.

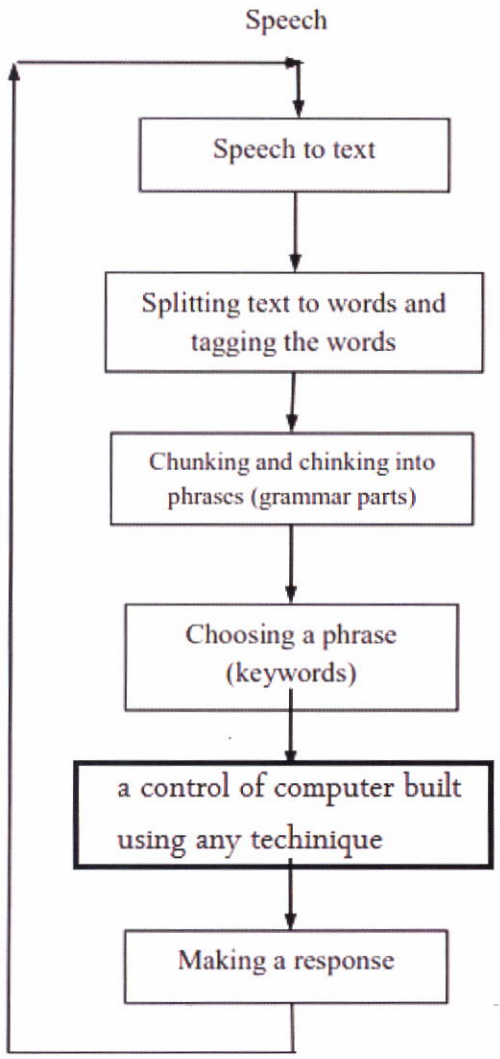

Fig2.4. The main steps of analysis and processing to perform human computer conversation

2.5 Syntactic Analysis for the Sentence

To analyse a sentence using NLP, word tokenising is needed for the sentences in order to split the sentence into individual words. Then each word should be POS tagged with a part of speech to identify the type of each word in the sentence. NLTK-POS tagger tags the first word in the sentence as NNP, which means a proper name, or PRP which means a

pronoun, assuming that the first word in a sentence must be a subject according to basic English language rules. For example:

- Beckham is joining the Los Angeles Galaxy after years with Real Madrid. is POS tagged as follows:

[('Beckham', 'NNP'), ('is', 'VBZ'), ('joining', 'VBG'), ('the', 'DT'), ('Los', 'NNP'), ('Angeles', 'NNP'), ('Galaxy', 'NNP'), ('after', 'IN'), ('years', 'NNS'), ('with', 'IN'), 'Real', 'JJ'), ('Madrid', 'NNS')]

- It is not that easy.

is POS tagged as follows: [('It', 'PRP'), ("is", 'VBZ'), ('not', 'RB'), ('that', 'IN'), ('easy', 'JJ')]

NLTK-POS tagger does the same thing with the sentences that begin with a verb. It tags the verb which sits in the beginning of the sentence as NNP, which makes it indistinguishable from the nouns or pronouns. For example:

- Give me a football.

[('Give', 'NNP'), ('me', 'NNP'), ('a', ' DT '), ('Football', 'NNP')]

We hard coded in Python programming language to tag the verb in the beginning of a sentence as a verb. The sentence in example 12 above can be POS tagged as follows: 13. [('Give', 'VB'), ('Me', 'NNP'), ('a', ' DT '), ('Football', 'NNP')]

2.6 control computer using speech recognition [8]

Speech recognition can be formally defined as the process of converting spoken words to text. That is how our phones and other devices can understand what we are saying, by converting our voice into strings and then processing it.

Speech is the most basic means of adult human communication. The basic goal of speech processing is to provide an interaction between a human and a machine.

Speech processing system has mainly three tasks –

- **First**, speech recognition that allows the machine to catch the words, phrases and sentences we speak
- **Second**, natural language processing to allow the machine to understand what we speak, and
- **Third**, speech synthesis to allow the machine to speak.

This topic focuses on **speech recognition**, the process of understanding the words that are spoken by human beings. Remember that the speech signals are captured with the help of a microphone and then it has to be understood by the system.

A- Building a Speech Recognizer

Speech Recognition or Automatic Speech Recognition (ASR) is the center of attention for AI projects like robotics. Without ASR, it is not possible to imagine a cognitive robot interacting with a human. However, it is not quite easy to build a speech recognizer.

B- Difficulties in developing a speech recognition system

Developing a high quality speech recognition system is really a difficult problem. The difficulty of speech recognition technology can be broadly characterized along a number of dimensions as discussed below –

- **Size of the vocabulary** – Size of the vocabulary impacts the ease of developing an ASR. Consider the following sizes of vocabulary for a better understanding.

- A small size vocabulary consists of 2-100 words, for example, as in a voice-menu system
- A medium size vocabulary consists of several 100s to 1,000s of words, for example, as in a database-retrieval task
- A large size vocabulary consists of several 10,000s of words, as in a general dictation task.

Note that, the larger the size of vocabulary, the harder it is to perform recognition.

- **Channel characteristics** – Channel quality is also an important dimension. For example, human speech contains high bandwidth with full frequency range, while a telephone speech consists of low bandwidth with limited frequency range. Note that it is harder in the latter.
- **Speaking mode** – Ease of developing an ASR also depends on the speaking mode, that is whether the speech is in isolated word mode, or connected word mode, or in a continuous speech mode. Note that a continuous speech is harder to recognize.
- **Speaking style** – A read speech may be in a formal style, or spontaneous and conversational with casual style. The latter is harder to recognize.
- **Speaker dependency** – Speech can be speaker dependent, speaker adaptive, or speaker independent. A speaker independent is the hardest to build.
- **Type of noise** – Noise is another factor to consider while developing an ASR. Signal to noise ratio may be in various ranges, depending on the acoustic environment that observes less versus more background noise –
  - If the signal to noise ratio is greater than 30dB, it is considered as high range
  - If the signal to noise ratio lies between 30dB to 10db, it is considered as medium SNR
  - If the signal to noise ratio is lesser than 10dB, it is considered as low range

For example, the type of background noise such as stationary, non-human noise, background speech and crosstalk by other speakers also contributes to the difficulty of the problem.

- **Microphone characteristics** – The quality of microphone may be good, average, or below average. Also, the distance between mouth and micro-phone can vary. These factors also should be considered for recognition systems.

Despite these difficulties, researchers worked a lot on various aspects of speech such as understanding the speech signal, the speaker, and identifying the accents.

Finally, we explored Speech Recognition with Python. We used the Speech Recognition package and saw how we can get information from the microphone. Now, we are able to clean up the signal as well in case of a noisy background. It is amazing how easy it is to use technology that was science fiction just a few years back. We are even able to use different languages. This technology is already heavily used for home automation and IOT problems, and many other applications are right around the corner.

2.7 NLTK Part of Speech Tagging

Once you have NLTK installed, you are ready to begin using it. One of the more powerful aspects of NLTK for Python is the part of speech tagger that is built in. Notably, this part of speech tagger is not perfect, but it is pretty darn good. If you are looking for something better, you can purchase some, or even modify the existing code for NLTK.[7]

The idea of part of speech tagging is so that you can understand the sentence structure and begin to use your program to somewhat follow the

meaning of a sentence based on the word used, its part of speech, and the string it creates.[7]

2.8 Python - Overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

2.8.1 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages,

including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

2.8.2 Python Features

Python's features include –

1. Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
2. Easy-to-read – Python code is more clearly defined and visible to the eyes.
3. Easy-to-maintain – Python's source code is fairly easy-to-maintain.
4. A broad standard library – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
5. Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
6. Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
7. Extendable – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
8. Databases – Python provides interfaces to all major commercial databases.

9. GUI Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
10. Scalable – Python provides a better structure and support for large programs than shell scripting.
11. Apart from the above-mentioned features, Python has a big list of good features, few are listed below –
12. It supports functional and structured programming methods as well as OOP.
13. It can be used as a scripting language or can be compiled to byte-code for building large applications.
14. It provides very high-level dynamic data types and supports dynamic type checking.
15. It supports automatic garbage collection.
16. It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

CHAPTER THREE

PRACTICAL ASPECT

3.1 Introduction

The practical part of the system is created using Python language and NLTK library is used for the sentence splitting process into words, speech library is used to convert speech into text As well as the Numpy library and the aiml is used to create the database.

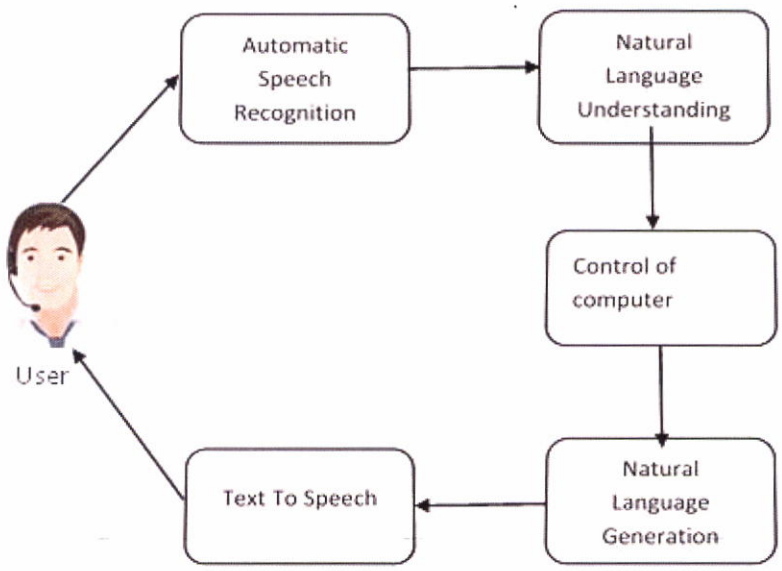

Figure (3.1) diagram of the system

As seen in the figure (3.1) how the system work and to read and cut text and perform the required according to the input by the user

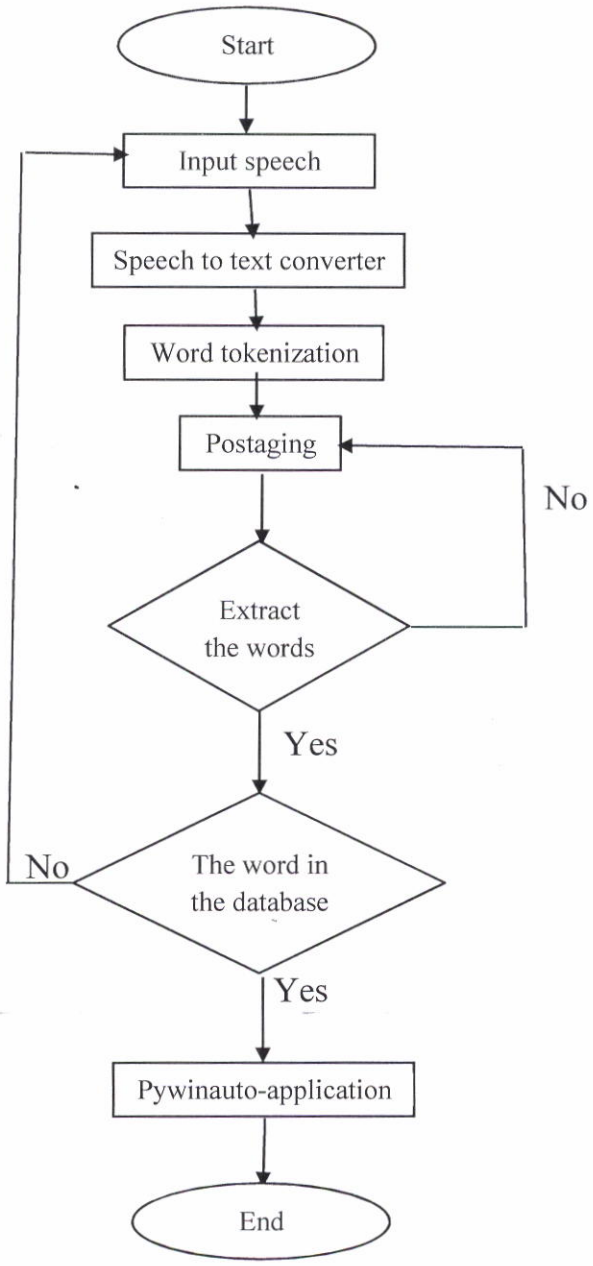

Figure (3.2) The flow diagram of the implemented system

As illustrated in the flowchart above, our system consists of several parts that are explained in brief:

(Input speech) In the beginning, the process of entering the recording by the microphone and then comes the process of **(Speech to text converter)**, which includes the process of converting speech to readable text and ready for the process of entering the system for a process to be

cut by the step **(Word tokenization)** This step involves the use of the NLTK library, which includes the process of cutting speech into words of each word, and then comes the **(postaging)** part of speech tagging which includes understanding the structure of the sentence and start using your program to follow the meaning of the sentence to a certain extent based on the word used then comes the process of **(extracting the words)**, which include extracting the words that are understood by the postaging process then comes the **(word in the database)** step, which includes the process of checking the resulting words if they are in the database or not, if they exist will be transferred to the next step and if not exist will return the process to step input speech Finally comes the step of **(pywinauto-application)** which includes the process of conducting the required depending on the words coming from the database, for example open the Word program or turn off the computer and so on.

The system consists of the following interfaces:

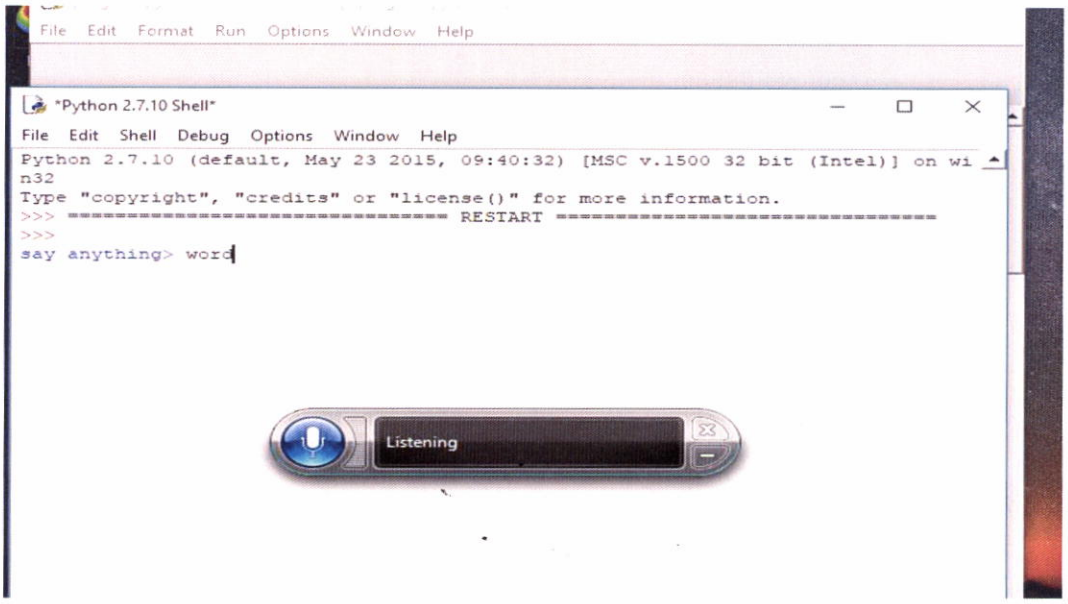

Figure (3.3) the main interface of implementation

As in Fig (3.3) we notice how the system understands the words entered and do the required work.

For example, if we wanted to call the Word Office program, we would write the Word and the system will call the Word program as we see in figure (3.4),(3.5). All these operations are a storage operation in the database.

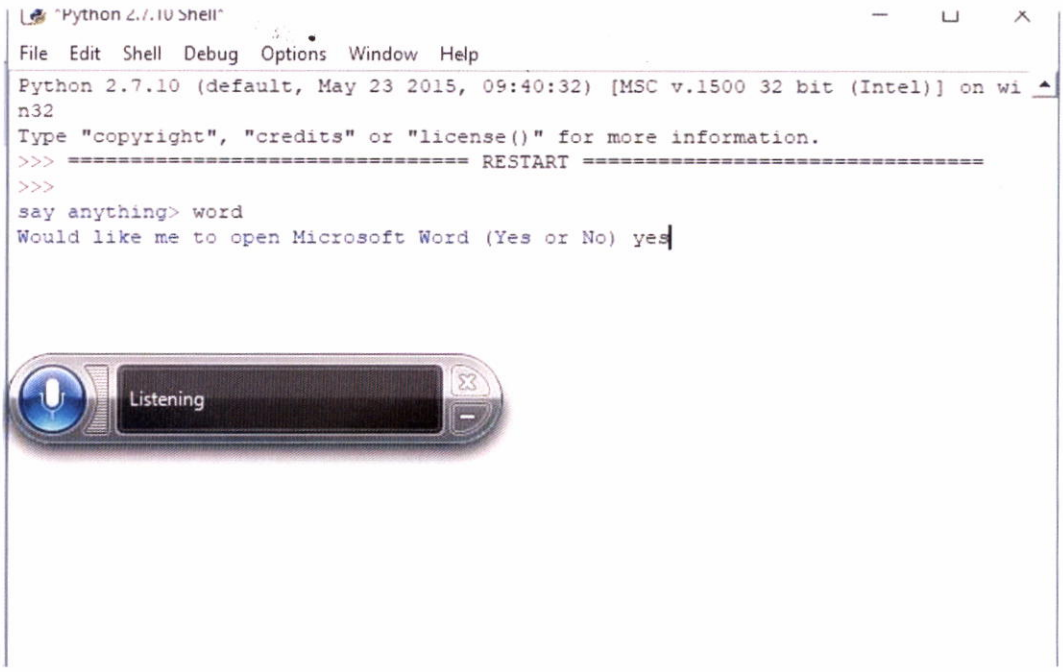

Figure (3.4) implementation of opening word program

As In figure (3.4) the system will call would like me to open Microsoft word , if we call yes , the system will open Microsoft word In figure (3.5).

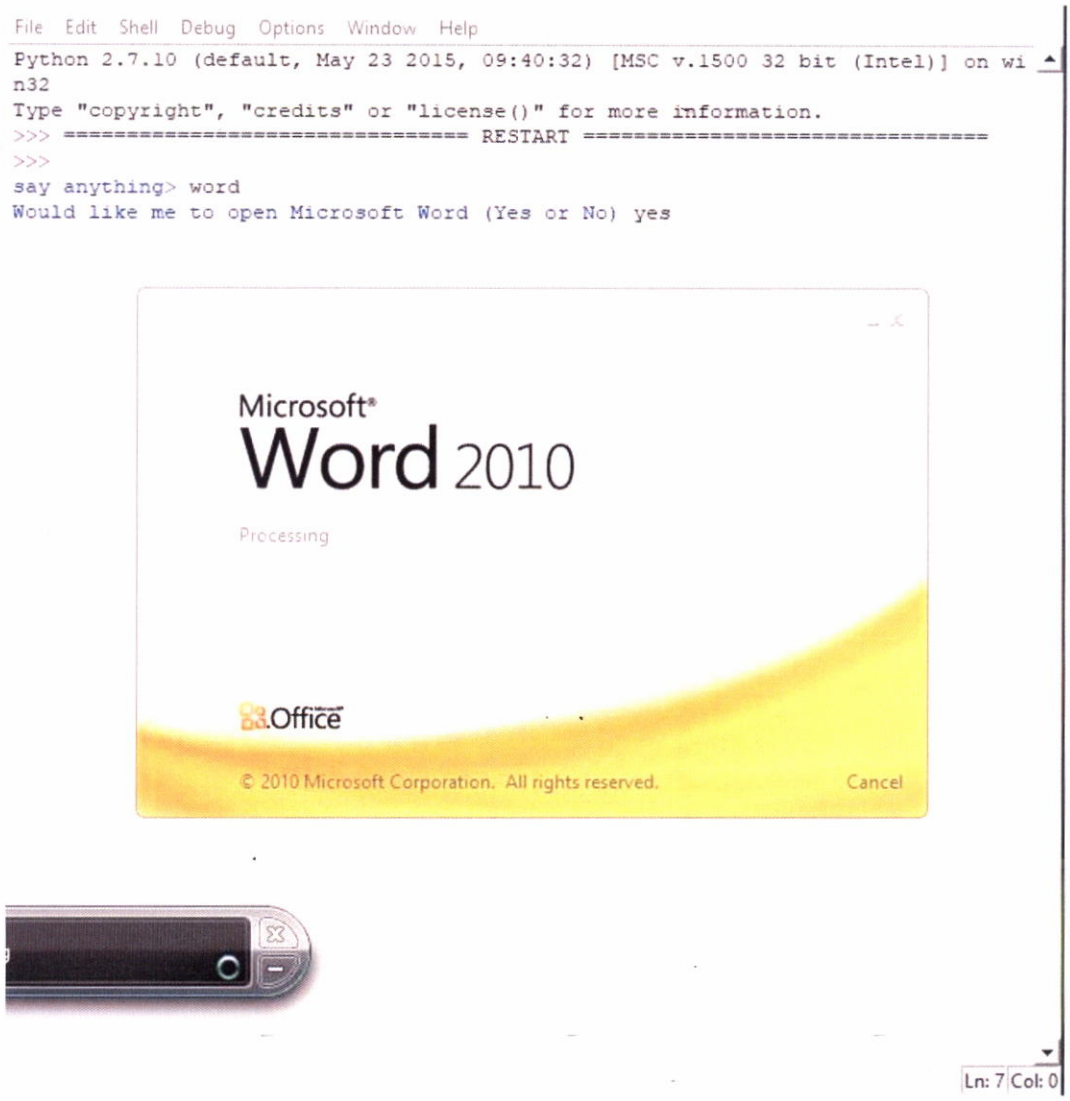

Figure (3.5) opening the word program

As in In figure (3.5) the Microsoft word is opened.

Another example is opening the internet explorer program as we see in figure (3.6), (3.7) explorer program

```
*Python 2.7.10 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on wi
n32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
say anything> net
Would like me to open Internet explorer (yes or no)
```

```
*Python 2.7.10 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on wi
n32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
say anything> net
Would like me to open Internet explorer (yes or no) yes
```

Figure (3.6) calling the internet explorer program

As in figure (3.6) if we wanted to call net , we would write net and the system will call would like me to open internet explorer. if we call yes. , the system will open internet explorer program In figure (3.7).

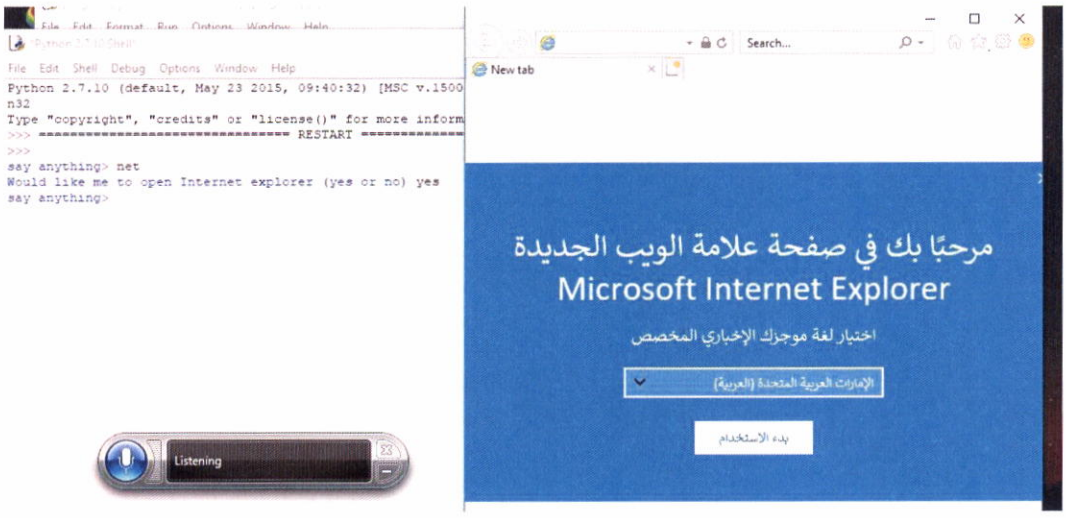

Figure (3.7) opening the internet explorer program

As in In figure (3.7) internet explorer program is opened.

Another example is restarting the computer as we see in fig (11) , (12)

```
*Python 2.7.10 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on wi
n32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
say anything> restart
```

```
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on wi
n32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
say anything> restart
Would like me to open again the computer(yes or no) yes
```

Figure (3.8) calling restart

As in In figure (3.8) if we wanted to call restart, we would write restart and the system will call would like me to open the computer again. if we call yes. the system will restarting the computer In figure (3.9).

Figure (3.9) restarting the computer action

Finally, we have created a system capable of reading the speech entered by the user and converting the speech to text and the process of cutting this speech and the process of correction of the words entered and do what is required depending on the final text.