Evaluation methodology for Speech To Text Services similarity and speed characteristics focused on small size computers

J.E. Aguilar-Chacon$^1$ and D.A. Segura-Torres$^1$

$^1$Facultad de Ingeniería Electrónica, División de Ingenierías, Universidad Santo Tomás, Carrera 9 No. 51-11, Bogotá, Colombia

jorgeaguilar@usantotomas.edu.co, dariosegura@usantotomas.edu.co

Abstract. The currently small size computers use and the high tendency to use human – voice interfaces to interact with any systems lead to merge these technologies. But implements strong Speech To Text algorithms on a small micro controlled system will reduce its computational capacity getting lower the system performance to execute other tasks. This work presents some selection criteria and a methodology to evaluate the Speech To Text services, using similarity and speed as metrics and the results obtained from an experiment deployed on a Raspberry Pi 3 using the proposed methodology with Spanish speakers. The application of the proposed methodology helped to detect the Google’s STT system as a best option over IBM and Microsoft STT systems because shows a faster conversion and better similarity measures.

1. Introduction

This work was developed to find the better option to implement a Speech To Text (STT) system for Spanish speakers into a small size computer like Raspberry Pi. Before to explore the wide world of the STT systems the possibilities were limited by the following criteria. First, to reduce the development time was decided to find a STT API. Second, there are two types of STT systems according to the internet connection needs approach, some systems need an internet connection, called online, and other didn’t, called offline. The offline STT systems process the audio information into the micro-controlled system, these has a program to capture, analyze and process audio signals. These systems manage all information locally, therefore, the system should be trained before their use, and a dictionary must be fed with new words to improve the STT system performance. The online STT systems usually invokes a web service, the audio is locally captured, but the audio is processed on remote server, the system on the server is trained by every audio processed, and many collaborators around the world feeds its dictionary. In this way, the online STT systems are better than the offline because the online systems are more trained and have a bigger dictionary, both properties give to the online STT systems a wide recognition range, can recognize more speakers and return a more precisely text. Then to get better results was selected an online STT System. Also, an online STT system reduce the computational load to the small size computer [1], because the process is made on a remote machine. Finally, the options were reduced by select the three most STT systems commercially used, these are: Google Web Speech API, Watson, and Azure Speech Service.
¿Which of the three is the best Speech To Text System to be implemented on a small computer like Raspberry Pi with Spanish speakers? to answer this question a methodology was proposed, an experiment has done using the proposed methodology, the experiment was developed over a Raspberry Pi and Python as programming language. Two metrics was selected to measure and compare the STT systems.

The following, it showed the proposed methodology, the work development and the obtained results by the experiment.

2. About commercial Speech To Text Services
In the field of technology there is a wide variety of voice-to-text services, the most commercially used are developed by recognized software companies. These services are called "Google Web Speech API" created by Google, "Watson" created by IBM and "Azure Speech Services" created by Microsoft. Python has a library called "SpeechRecognition", this library supports different APIs for speech recognition and audio to text conversion, including Google, IBM and Microsoft Services. These APIs provide real-time speech to text conversion, allowing systems to implement features such as voice commands, conversation transcripts and call centers. In addition, the voice-to-text services mentioned above have the ability to adapt to a different voice accent or voice pattern without changing the text resulting from a voice conversion.

"Google Web Speech API" and "Google Cloud Speech" are APIs implemented by some web pages, home automation systems and other devices that use voice recognition. The accuracy of the APIs is increasing due to the large neural networks that support them, the improvements in each version and the updates for each language and region.

"Google Web Speech API" is not recommended for an arduous production environment, its use is limited to fifty requests per day and it is not possible to increase this number of events, but "Google Cloud Speech" allows you to create a trial version that gives sixty minutes of audio conversion, or creates a paid account. When purchasing any of the packages offered by the company, the package will have features such as automatic speech recognition, more than one hundred and twenty languages adapted to different regions, support for pre-recorded files or live stream, filters for different noise conditions or inappropriate vocabulary, and selection of deployment models.

"Azure Speech Services" is the Microsoft API cloud service that allows the conversion from voice to text. In contrast with "Google Web Speech API", "Azure Speech Services" has two integration options, a device can use the Software Development Kit (SDK) or the REST-API. The SDK allows an easy way to integrate "Azure Speech Services" into a device, using traditional programming languages such as C# or JAVA. "Azure Speech Services" offer the protection of captured data and integration with other language services like cognitive services provided by Microsoft. This API has a limitation when using the account, the free version lasts sixty days, after this time, you must make a paid subscription.

The IBM Speech to Text service, better known as "Watson Speech to Text" has conversion time limitation like Google’s, with this service it is possible convert one hundred minutes audio, to increase this time it is necessary to realize a payment subscription.

3. Proposed methodology
The selected metrics for the methodology, the task that conform it, and the algorithm used to implement the methodology are described in this section.

3.1. Metric used to evaluate speed and similarity
The currently experiment aims to know the faster and more reliable Speech To Text commercial service for male and female speakers, this way, the experiment will measures the conversion time for each of three systems to convert an audio and the percentage similarity comparing the original text and the output text of the STT system.
The conversion time is measured with a difference between two timestamps, one taken just before calling a service and the second when the service returns the text.

The similarity between two text strings is determined with Levenshtein's algorithm. This is a dynamic programming algorithm devised by Vladimir Levenshtein in 1965. Levenshtein's distance is a measure of similarity between two text sentences; this measure is the number of elementary operations (insertion, substitution, and deletion of a character) performed to the original string to obtain the target string. The number of operations performed indicates the similarity between the two words, so if Levenshtein's distance is very large, it means that the two texts are very different [2]. Then the similarity precision percentage is the ratio between the Levenshtein’s distance and the length of the longest of the two analyzed text strings.

### 3.2. Proposed methodology

The usual flow for a Speech To Text system has the following steps [3-5], see **Figure 1**, first the speaker talks or reads in front of a microphone, this audio is taken by a digital system using an analog to digital converter, the digital system processes the audio using an algorithm, and the algorithm returns a text. This flow represents some problems to compare two STT systems.

The first problem is the speaker training, that means the speaker will improve his reading skills while the experiment is repeated, the second reading of a text made by someone will be better than the first read, therefore repeat an experiment many times with a speaker will falsely improve the result on a STT system. The second problem is the experiment repeatability, it is about to repeat a STT experiment under the same conditions with a speaker will be difficult, because in the second iteration the speaker may be tired, the environment noise will change, the speaker cannot be available for the experiment all the time, and other considerations.

![Figure 1](flow.png)

**Figure 1.** Flow for a Speech To Text system.

Record the speaker voice when reading once is the solution for these problems, the speaker read or speak only one time, this reading is recorded and the audio can be processed many times, giving to the digital system always the same information.

Then, the experiment methodology selects a text, record some female and male voices reading the text, process each voice file many times, store the measured results for conversion time and percentage similarity, then these results are analyzed by hand.

The selected text was taken from the fairy tale “Pinocho” because it is well known for the speakers and do not have difficult words, the text has 39 word divided in two sentences. The experiment uses Spanish speakers, 4 female and 6 male different voices were recorded into a file using a standard cellphone. The files were processed by an algorithm into a microprocessed system, this algorithm executes STT algorithm, calculate the metrics and stores the results. Finally, the information was analyzed using a commercial spreadsheet.

### 3.3. Algorithm for the methodology

As mentioned before, the methodology has steps made by hand; these are capture voice and information analysis. Then, the automatized steps are the reading voice file, the calling STT service, getting the identified text from the STT, calculate the time taken by the service and the percentage, and finally store these results.
The developed algorithm allows automatically process many files into selected STT service or SST system. The STT algorithm flowchart used is showed in the Figure 2. The files to be processed should be copied into specified folder into the system. A list with the files names is preloaded by code, the system loads the first file in the list. Just after loading the audio an initial time mark is stored and the audio file is sent to the selected STT service. This STT system will begin to perform the conversion of the audio and at the end of its process return the identified text, at this point a final time mark is added, the value of the difference between the final mark and the initial mark will indicate the total time taken by a system to perform the speech to text conversion, the next step is obtain the text returned from the system, this text is compared with the original text using the Levenshtein's distance and calculating the similarity precision percentage.

The time and precision data obtained are stored in a CSV file with additional information like date and time experiment, file name, and STT service name. The system reads the next file on the list and repeat the process until file names on the list was ended. In order to perform the cycle again with a different system, the experiment is reloded but calling a different STT system and storing these results.

![STT Algorithm Flowchart](image)

**Figure 2.** STT Algorithm Flowchart.

3.4. **Comparison with previous work**

Prior to experimental development, a bibliographic consultation on similar experiments was carried out. The most important difference is the language used in the tests, for this experiment the language is Spanish, in other experiments the language used is English.

The methodology used for the comparison of different recognition systems in the article "Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx)" [6] does not use the time variable as a metric, and they have another algorithm different from "Levenshtein's distance" for the comparison of the received audios with respect to the original audios, another difference is the implementation of different sentences for the tests, we only use the same text but recorded by different people. In another article " A Comparison of Online Automatic Speech Recognition Systems and the Nonverbal Responses to Unintelligible Speech" [7], a difference was found with respect to the methodology used for the experimental phase, the methodology used for the insertion of the audio to the different conversion services is manual, this methodology can be convenient if the experiment is carried out only once, but if in the experiment several tests are needed, it is better to automate the process and create an algorithm.
4. Experimental tests for the proposed methodology

4.1. Environment and system description
Small computers have become a useful ally for those who in recent years have decided to be immersed in the world of development, learning and use of new technologies for different functionalities. These devices have many specifications and services which can be used to implement an application according to the system requirements. This experiment uses Raspberry Pi 3 board, the most successful minicomputer in the market, because its powerful processor, different ports and connections, and a good purchase price in the market. Some specifications for the used board are listed in Table 1.

| Specifications | Raspberry Pi 3 Specifications | Raspberry Pi 3 |
|----------------|--------------------------------|----------------|
| Price          | $35 USD                        | HDMI           |
| System on Chip | BCM2837                        | 4xUSB          |
| Manufacturer   | Broadcom                       | 10/100 Ethernet|
| CPU            | ARM                            | 40 GPIO pins   |
| Instructions   | ARmv8 64bits                   | MIPI Camera    |
| Cores          | Quad-Core                      | Connector      |
| Speed          | 1200MHz                        | MIPI Display DSI|
| RAM            | 1GB (1024MB)                   | Composite video|
| Storage        | MicroSD Slot                   | (PAL and NTSC) |
| GPU            | 400MHz                          | via 3.5mm TRRS jack|
| Power supply   | 5V, 3A                         | shared with stereo audio |
| OS             | Raspbian V4.19                 | Wi-Fi 2.4GHz   |

The algorithm was implemented in using Python scripts. For the experiment, it is necessary to install certain libraries, libraries that allow updating the system to its latest version and the inclusion of libraries for microphone uses and speech recognition. In addition, an audio coding command was installed and all the requirements for the proper functioning of the STT systems. Finally, the "Speech Recognition" library was installed, this library implements the voice recognition interfaces to invoke some STT services or STT systems.

Commands used in the Raspberry Pi console application to get the system ready are:
- `sudo apt-get update`
- `sudo apt-get install python`
- `sudo apt-get install python-pyaudio python3-pyaudio`
- `sudo apt-get install portaudio19-dev python-all-dev python3-all-dev & & sudo pip install pyaudio`
- `sudo apt-get install flac`
- `pip install SpeechRecognition`
The "SpeechRecognition" library is installed by default with "Google Web Speech API", this API is ready for testing. The installation process was done following the instructions of a guide found on the book called Building a virtual assistant for Raspberry Pi: The practical guide for constructing a voice-controlled virtual assistant, where the process is explained step by step. [8-9]

5. Results

After runs the experiment using the 10 recorded files, using three systems, and repeated it several times, the average conversion times are showed in the Table 2. The faster system was Google’s, and IBM’s was the slowest. The difference between Microsoft’s and Google’s takes around 33.7% less time than Microsoft’s STT system, and around 48% less time than IBM’s, its means significative differences between IBM’s system and Google’s.

Table 2. General average of time, with respect to STT systems.

| STT systems | Average conversion time (mS) |
|-------------|------------------------------|
| Google      | 6101,579                     |
| Microsoft   | 9205,645                     |
| IBM         | 11744,616                    |
| Grand Total | 9017,279822                  |

The results on the ¡Error! No se encuentra el origen de la referencia., shows the average time by STT system and gender, all the engines perform the conversions in similar time for both genders, but curiously for all STT systems the female voice conversions takes less time than male voice, but it requires other experiment or different studies beyond this work.

Table 3. Average conversion time respected to genders and STT systems.

| Average conversion time | Gender            |
|-------------------------|-------------------|
|                         | Male Voice (mS)   | Female Voice (mS) | Grand Total (mS) |
| Google                  | 6592,209          | 5365,630          | 6101,579         |
| Microsoft               | 9308,767          | 9050,961          | 9205,645         |
| IBM                     | 12141,361         | 11149,497         | 11744,616        |
| Grand Total             | 9347,446          | 8522,031          | 9017,280         |

According to the Table 4, the best similarity percentage was achieved by Google’s service, but Microsoft’s and IBM’s services are very similar. Nevertheless, all services are over the 90% similarity percentage.

Table 4. General average of precision in percentage, with respect to STT systems.

| STT Systems | Similarity percentage |
|-------------|-----------------------|
| Google      | 97,116%               |
| Microsoft   | 90,825%               |
| IBM         | 90,337%               |
| Grand Total | 92,759%               |
Also for the percentage similarity, the STT systems are gender independent, according to the results on the Table 5.

| Similarity percentage | Gender          |
|-----------------------|-----------------|
| STT Systems           | Male Voice      | Female Voice | Grand Total |
| Google                | 96,566%         | 97,940%      | 97,116%     |
| Microsoft             | 90,893%         | 90,722%      | 90,825%     |
| IBM                   | 91,056%         | 89,258%      | 90,337%     |
| Grand Total           | 92,838%         | 92,640%      | 92,759%     |

6. Conclusions
The conclusions obtained after applying the proposed methodology are:

- The proposed methodology allows the experiment repeatability with just one reading of speaker.
- With the proposed methodology the experiment can be replicated several times in a short time.
- According to the selected metrics, Google’s STT system is the faster and presents the higher percentage similarity compared with Microsoft’s and IBM’s systems.
- According to the obtained results, the STT services are gender voice independent for the percentage of similarity.
- The used STT services perform faster conversion with female voices than male voices but is necessary a deeper test to confirm or deny this conclusion.

References
[1] Rosales J, Castillo J, Mansilla F and Boza G 2014 A Prototype of a Semantic Platform with a Speech Recognition System for Visual Impaired People (Journal of Intelligent Learning Systems and Applications) pp 87-92.
[2] Ene A and Andrei E 2017 An application of Levenshtein algorithm in vocabulary learning (2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)) pp 1-4.
[3] Kishore, Kumar M, J Jayalakshmi and S Karthik 2018 A Python base Virtual Assistant using Raspberry Pi for Home Automation (SSRG International Journal Of Electronic And Communication Engineering) pp 22-27
[4] Al-Amin Md and Syeda Zinath Aman 2016 Design of an Intelligent Home Assistant (2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)) pp 69-71.
[5] Surinder, Kaur, Sanchit S, Utkarsh J and Arpit R 2016 Voice Command System Using Raspberry Pi (Advanced Computational Intelligence: An International Journal (ACII)) pp 43-49.
[6] Kepuska, Veton and Gamal Bohouta 2017 Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx) (Journal of Engineering Research and Application) pp 20-24.
[7] Kim Y, Joshua and others 2019 A Comparison of Online Automatic Speech Recognition Systems and the Nonverbal Responses to Unintelligible Speech (AiXiv) pp 1-13.
[8] Pant T 2016 Building a virtual assistant for Raspberry Pi: The practical guide for constructing a voice-controlled virtual assistant (Berkeley: Apress) volumen I
[9] Pedraza J, Ramos J, Vargas J, Aceves M and Martinez L 2018 Prototipo de Asistente Virtual para su Aplicación en Edificios Inteligentes (Robotica y Mecatronica) chapter 4 pp 37-46