ISSN: 2230-9926

Available online at http://www.journalijdr.com

IJDR

International Journal of Development Research
Vol. 11, Issue, 11, pp. 52248-52256, November, 2021
https://doi.org/10.37118/ijdr.23437.11.2021

RESEARCH ARTICLE OPEN ACCESS

OFFLINE CONTINUOUS SPEECH RECOGNITION FOR MOBILE DEVICES

Lucas Debatin[1], Aluizio Haendchen Filho[1*], Rudimar Luís Scaranto Dazzi[1], Hércules Antonio do Prado[2] and Edilson Ferneda[2]

[1]Laboratory of Applied Intelligence, University of the Itajaí Valley, Itajaí-SC, Brazil
[2]Catholic University of Brasilia, Brasília-DF, Brazil

ARTICLE INFO

Article History:

Received 10th August, 2021
Received in revised form
12th September, 2021
Accepted 24th October, 2021
Published online 30th November, 2021

Key Words:

Continuous Speech Recognition, Offline Speech Recognition, Natural Language Processing, Mobile Devices.

****Corresponding author:*
*Aluizio Haendchen Filho***

ABSTRACT

Speech recognition is an option of accessibility in electronic devices. Currently, this recognition is accomplished through Application Programming Interfaces that depend on Internet connection and which are often made available by proprietary software. Voice recognition is an important technology for improving HCI (Human-Computer Interaction), after all speech is a human feature that most people have. The increased use of adaptive interfaces using speech recognition results from the fact that speech is the most natural form of interaction. Speech makes it faster to access information in software and applications, compared to standard interaction forms such as touchscreen, mouse, keyboard, among others. Considering this context, this work presents an open-source solution for continuous speech recognition in mobile devices. At first, a tool comparing the processing and memory usage of library configurations on a desktop computer was developed. Then, we implemented the best alternatives in an Android application and tested its performance and battery usage on several mobile devices. For this, the best libraries were selected, and their configuration files were optimized to find the most cost-effective solution between performance and accuracy of the Word Error Rate (WER) and Sentence Error Rate (SER) rates.

Citation: ***Lucas Debatin, Aluizio Haendchen Filho, Rudimar Luís Scaranto Dazzi, Hércules Antonio do Prado and Edilson Ferneda.*** "Offline Continuous Speech Recognition for Mobile Devices.*", International Journal of Development Research*, 11, (11), 52248-52256.

INTRODUCTION

Since the emergence of computers, researchers have been looking for ways to make computer systems smarter. One of these forms is the understanding of speech, which aims to make machines capable of understanding and communicating in natural language. To understand speech computationally, it is necessary to convert spoken language into text; this process is called speech recognition (Soni, 2019; Jurafsky and Martin, 2008). The increased use of adaptive interfaces using speech recognition results from the fact that speech is the most natural form of interaction. Speech makes it faster to access information in software and applications, compared to standard interaction forms such as touchscreen, mouse, keyboard, among others (Hearst, 2011). Voice recognition is an important technology for improving HCI (Human-Computer Interaction), after all speech is a human feature that most people have. Advances in speech recognition techniques have enabled the use of this technology in many applications, including mobile devices. While many tasks are best addressed with visual interfaces (keyboard, mouse, and so on), voice has the potential to be a more natural interface. It can provide interaction even if the user is busy with his hands and eyes or haves limited abilities. These are advantageous regardless of the type of user, except for people with aphonia or dysphonia (NationalAcademies of Sciences, Engineering, and Medicine, 2017; Yu and Deng, 2015). Speech recognition can be classified into two types: *(i)* Isolated words, which require sentences to be pronounced with pauses between each word, useful in simple systems and with small vocabulary, such as, command and control systems by voice; and *(ii)* Continuous, focusing at making communication more effective for humans, since they recognize sentences pronounced in a natural way, that is, without the need for pauses between words (Huang and Deng, 2010). Our approach works with continuous type recognition. This mode of speech recognition is more complex when compared to the recognition of isolated words, since it's able to handle all the characteristics and vices of speech. Moreover, in this type, there is no information on where certain words or phonemes begin and end. Consequently, many words can be replaced or unidentified (Jurafsky and Martin, 2008). Currently, Web Speech, Java Speech, Google Cloud Speech and Bing Speech APIs facilitate the implementation of continuous voice recognition of Brazilian Portuguese in software and applications. However, these APIs cannot be used in all application types of application because they need an internet connection. In addition, they are proprietary software, can cash in on access, and generate cost for those who use it on a large scale (Neto et al., 2011; Tevah, 2006; Debatin et al., 2018). Although it characterizes an old approach, since society is currently in the cloud computing age, the offline technique has some advantages: *(i)* it does not suffer from problems related to latency and bandwidth, since they do not need servers in the cloud or remote; *(ii)* do not have server-side sharing issues, since cloud services serve multiple clients, and *(iii)* do not present security, compliance, and regulatory issues (Grossman, 2009). In order to implement speech recognition, it is necessary to know some of its fundamental properties, such as: *(i)* the features of the voice signal, which are the various information about the speaker present in the audio signal; *(ii)* the forms for extracting and decoding audio characteristics, used to generate the best textual sequence from the input audio signal; and *(iii)* the evaluation metrics, which are used to measure speech recognition performance (Ferreira and Souza, 2017). To enable recognition in offline mode, it is necessary to perform voice recognition processing on the mobile device itself. Voice recognition has a high computational complexity and requires a lot of memory, and this is a constraint, since many smartphones and tablets have limited hardware resources (Georgescu et al., 2021; Gupta and Owens, 2011). This work presents a solution for off-line continuous speech recognition. After a systematic review of the literature, the libraries CMUSphinx, Hidden Markov Models Toolkit (HTK) and Kaldi were selected and analyzed. Each library has specific configuration files that were edited to find the most cost-effective configuration of Word Error Rate (WER) and Sentence Error Rate (SER) performance and accuracy. A tool was developed to analyze the procedures, comparing the processing and memory usage of the library settings on a desktop computer. The better evaluated library and configuration were selected and implemented in an Android application, that was evaluated for the sake of performance and battery usage.

MATERIALS AND METHODS

This section presents the fundamentals that guided this investigation along with technical references adopt to implement the solution.

BACKGROUND

The basis for discussing speech recognition, involving the concepts, the libraries, and the speech corpus considered in this work are presented in this subsection.

Speech Recognition: Speech recognition is the process of converting the analog speech signal into its textual representation. In this way, the generated text is composed of the sequence of words that have been identified from the input signal. Speech recognition is a research area that has been active for over five decades. In the old days, the speech was little used in the HCI, since it did not surpass the efficiency and precision of the keyboard and the mouse. This is due to the fact that the technology of the time was not advanced enough. However, in recent years the computational power has increased, allowing the training of larger and more complex models, thus reducing WER (Yu and Deng, 2015).

Speech recognition simulates the human hearing system, and its performance is influenced by the characteristics that affect the speech signal, such as the vocabulary used, characteristics of the speakers and background noises (Meyer et al., 2014).

The basic structure of speech recognition systems, shown in Fig. 1, is divided into two steps:

- Features Extraction: Applies algorithms to the speech signal of the input in order to represent it in a more compact and robust way. For this, it is necessary to convert the analog signal into a digital representation and determine what is silence/noise and to extract in fact the useful voice information. With the elimination of noise, the amount of voice information is reduced, consequently decreasing the computational cost (Serizelet al., 2017).
- Decoder: searches for the best sequence of words in a set of possible hypotheses given the representation of speech signal characteristics. This step uses the following models: *(i)* acoustic, which transforms the signal being processed into words and sentences; and *(ii)* language, which is responsible for characterizing the language and conditioning the combination of words discarding grammatically incorrect phrases (Ferreira and Souza, 2017).

The performance of speech recognition depends on the accuracy of the acoustic models, the complexity of the task defined by the language model and the quality of the acquired audio signal. In the literature there are several mathematical functions that can be called evaluation metrics. In this work the following will be evaluated: WER, SER and xRT (Yu and Deng, 2015; Ferreira and Souza, 2017). WER is one of the most used metrics in continuous speech recognition systems. The WER is based on the number of words the entered incorrectly, that were deleted, and that were replaced in comparison with the reference phrase. This rate is calculated by the equation below.

$$WER = \frac{S + I + R}{N}$$

where: *(i)N* is the total amount of words in the reference phrase; *(ii)S* is the total number of errors per substitution; *(iii)* I is the total amount of insertion errors; and *(iv)E* is the total amount of errors per exclusion of the generated phrase compared to the reference phrase. Fig. 2 shows an example of comparison between reference and generated sentences. As can be seen, the phrase generated has two substitutions, one inclusion and one exclusion, totaling four errors. The reference phrase has five words. By replacing the respective values in equation (1), the WER value is 0.8 (80%).

Reference = There	is	a	prisoner	in	the	*	cell
Generated = There	*	a	prisoner	on	the	to	sell
	E			**S**		**I**	**S**

The SER represents the number of sentences that have at least one error, that is, the number of sentences that have a WER higher than 0% (Ferreira and Souza, 2017). The SER is calculated using the following equation:

$$SER = \frac{E}{T}$$

where *E* is the number of sentences with at least one error and *T* is the total number of sentences. For example, in a corpus with 100 phrases, only 30 presented a WER higher than 0%. Thus, the SER in this example is 0.3 (30%). The xRT factor is used to calculate the speed of the speech recognition process. It is calculated by dividing the time the system spends to recognize a sentence by its duration, as shown in the equation below. The smaller is xRT factor, faster the recognition will be:

$$xRT = \frac{P}{D}$$

where *P* is the processing time spent to perform speech recognition of the audio file and *D* is the duration time of the audio file. For example, the computer takes 1.2 seconds to recognize a file of 28 seconds, so its xRT factor is 0.04.

Reference Libraries for Speech Recognition: One of the main reasons for using libraries for continuous speech recognition was to reduce time at the development process. However, it is necessary that they are constantly updated and also have a good documentation. Based on experimental results, the CMUSphinx, HTK and Kaldi libraries are the most commonly used. In the following subsections the characteristics of these libraries will be described.

CMUSphinx: Sphinx is a speaker-independent continuous speech library. It uses the Hidden Markov Models (HMM) technique in the acoustic model and the n-gram technique in the language model (Lee et al., 1990). The CMUSphinx tools are designed specifically for low-resource platforms and its license is similar to Berkeley Software Distribution (BSD), which enables commercial distribution. In addition, this library can be used for several purposes related to speech recognition, such as keyword identification, alignment, pronunciation evaluation, among others[1].

The CMUSphinx toolkit has several library packages for different tasks and applications. In this work, we used the PocketSphinx package that is ideal for embedded systems. Pocketsphinx is written in the C programming language and can be used with Linux, Microsoft Windows, MacOS, iPhone and Android.

HTK: HTK[2], available in the C programming language, is a toolkit for building and manipulating HMM. It is mainly used for speech recognition research, but it can be used in a variety of applications, such as voice synthesis, character recognition and DNA sequencing. In addition, the HTK tools provide sophisticated features for speech analysis, training, testing, and results analysis of HMM.

Kaldi: Kaldi[3] is an open-source speech recognition toolkit developed in the C++ programming language and licensed under the Apache License v2.0. Their tools compile on Unix and Microsoft Windows systems (Povey et al., 2011). The CMUSphinx and HTK libraries are Kaldi's main competitors, but these libraries do not have a finite-state transducer-based structure, broad support for linear algebra, and a non-restrictive license (Povey et al., 2011). In addition, Kaldi is the only library among the three that has Deep Neural Network (DNN) support, such as Multilayer Perceptron (MLP) and Recurrent Neural Network (RNN).

Speech Corpus: Corpora linguistic are collections of written or spoken language data that serve various types of research and can be used in all branches of linguistics. In other words, it may be argued that corpora are meant to represent a particular language as a whole. For example, a researcher might study a corpus of phone conversations to prove that people talk on the phone differently than when they talk in person (Bauer and Aarts, 2000). For languages that are different from English, like the Brazilian Portuguese, obtaining a large and free corpus is one of the main challenges encountered by researchers in the area (Ferreira and Souza, 2017).

In this work, the corpora available on the website of the FalaBrasil group were used, but these have few hours of duration. Most of the works published in Brazilian Portuguese are restricted to a reduced vocabulary. Therefore, the language model is constituted only by the phrases that are present in each speech corpus, that is, the recognition developed have a restricted vocabulary, since it recognizes only the words that are in the speech corpora(FalaBrasil, 2019; Silva et al., 2005). The LaPS Benchmark corpus consists of 700 phrases and has 35 speakers with 20 sentences each, 25 men and 10 women, corresponding to approximately 54 minutes of audio. All recordings were performed on computers using common microphones, and the environment is not controlled (presence of noise). The sampling rate used was 22050 Hz and each sample were represented with 16 bits (FalaBrasil, 2019). The corpus of Federal Constitution speech consists of 1,255 sentences with an average of 30 seconds each, totaling approximately 9 hours of audio with only one male announcer. The audio files were sampled at 22050 Hz with 16 bits. In addition, a controlled recording environment was used, with little presence of noise (FalaBrasil, 2019).

[1]https://cmusphinx.github.io
[2]http://htk.eng.cam.ac.uk
[3]http://kaldi-asr.org

It was necessary to split the speech corpora into training audio files and tests. The LaPS Benchmark corpus was divided into: *(i)* training with 30 speakers (640 files), where 23 male and 9 females were present; and *(ii)* tests with 3 speakers (60 files), two male and one female. The Federal Constitution corpus was divided into 1,129 files (90%) for training and 126 files (10%) for tests.

Proposed Solution: The libraries are free and have been installed on a computer with the Antergos operating system (Linux distribution based on Arch Linux) version 64 bits, according to the respective documentation. In the following subsections we will demonstrate the training implementation and the tool for performance analysis.

Training Implementation: Acoustic and speech recognition language models require training. In these libraries the supervised training technique is used, because they learn how to classify training data that have already been classified manually by humans (Ferreira and Souza, 2017; Copin, 2004). In total, 30 training sessions were created in each corpus, with 10 files with different configurations for each library. These 10 files were divided into two sets, modifying the configuration change logic, as highlighted below:

- In the first set, the default configuration values of all libraries were changed uniformly: *(i)* the values were reduced by 80%; *(ii)* the values were reduced by 40%; *(iii)* maintained the default setting; *(iv)* the values were increased by 40%; and *(v)* the values were increased by 80%. Thus, 15 configuration files were generated.
- In the second set, for each library we changed: *(i)*CMUSphinx, with the previously mentioned standard values and changing the textual options of the configuration parameters, for example, where "no" became "yes"; *(ii)* HTK, the default settings for extracting audio characteristics, using the same uniform change logic, -80%, -40%, 0%, 40%, and 80%; and *(iii)* Kaldi, values of -80%, -40%, 0%, 40%, and 80% of the configurations using DNN.
- Each configuration file is executed by the source library generating an output file. The purpose is to locate the best settings, with the following information: *(i)* start and end date and time; and *(ii)* the output of the library with the value of the evaluation metrics SER and WER.

Tool for Performance Analysis: The performance analysis tool developed in this project has two versions (desktop and mobile). They are used to measure the processor and memory usage of continuous speech recognition libraries, which have been implemented in Brazilian Portuguese. The name chosen for this tool was OCSR (Offline Continuous Speech Recognition). This tool includes the implementation of the source codes that are responsible for generating the output of the already trained models of the libraries, and only the files of test corpora were used.

Desktop Version: The desktop version was used to test the performance of libraries on a desktop computer. The result of this test was used to find the library that presented the best performance, which was implemented in the mobile version. The C++ programming language was used in conjunction with the Qt SDK (Software Development Kit). In this version we use the four training configurations with the best WER rate for each library (CMUSphinx, HTK and Kaldi). The implementation of the desktop version is available for free in GitLab[4]. Fig. 3 shows the screen of the desktop version, which has the following configuration options: *(i)* library, which is responsible for selecting the library (CMUSphinx, HTK and Kaldi) to be tested; and *(ii)* voice corpus, which is responsible for selecting the voice corpus (LaPS Benchmark and Federal Constitution) that will be tested. The "Test" button generates the result files for the selected library and corpus.

[4] https://gitlab.com/lucasdebatin/ocsr-desktop

For each library and corpus, three results files were generated: *(i)* start date and time; *(ii)* processor and memory usage, i.e. performance required; and *(iii)* closing date and time. The performance was captured using the Linux "top" command, which displays the data about the processes running on the device.

Mobile Version: The mobile version was used to test the performance on mobile devices of the library that performed best in the desktop version. The Kaldi library models presented the best results for corpora. However, the implementation of this in-app library was unsuccessful, as runnables only work on devices that are granted "root" permissions. Attempts to deploy Kaldi were based on the compilation tutorial from the Android library[5].

For this reason, only the CMUSphinx library models were used, which also presented excellent results compared to the results of the Kaldi library. The tests were only performed on mobile devices with the Android operating system, with version 4.1 being the minimum supported. For the implementation, the programming language Java and NDK (Android Native Development Kit) were used. The use of NDK made it easier to compile the C source code of the library. The application source code is available in GitLab[6]. Fig. 4 shows the application screen, and as you can see, only one message is displayed: *(i)* you are initializing the application by copying the models from the library to the internal memory of the device; and *(ii)* generating the test file, running the library with the test corpora speech files, and generating the results files. At the end of the tests, a text with the following results data is generated for each corpora: *(i)* percentage of the initial processor used to verify that the processor is not overloading with another process; *(ii)* the amount of available memory, the full memory indicator and the total amount of memory of the device; *(iii)* battery percentage at the beginning and end of tests; *(iv)* the start and end date and time; and *(v)* the use of the processor (maximum, minimum and average) and memory (maximum, minimum and average) during the execution of the tests.
The percentage of processor usage is obtained using: *(i)* the command "top" in versions smaller than the version 8 of Android; and *(ii)* the command "ps" in versions 8 and 9 of Android. The amount of memory and device battery is captured using Java libraries. It was not possible to capture the battery usage in watts during the execution of the library, for that reason only the percentage of use was captured.
For tests on mobile devices, it was necessary to reduce the amount of test files of the Federal Constitution corpus from 126 to 30, because in some devices the time spent for processing exceeded 30 minutes. This reduction was necessary to make the tests more attractive and quicker for the research volunteers. Additionally, the app has been made available on the Play Store[7] to facilitate installation in devices.

RESULTS

This section presents and discusses the results of the project, allowing an assessment of its contribution and the achievement of objectives. In the subsection 3.1, for each library, the training configurations with the best values obtained for the WER rate are presented. Subsection 3.2 presents the best performance of each library on a desktop computer. Subsection 3.3 shows the performance of the selected library on multiple mobile devices.

Training Settings: This section presents the evaluation metrics obtained for each configuration file. Based on the percentages obtained in the metrics, the best performance test configurations were chosen on a desktop computer. For the measurement of the SER and WER evaluation metrics, the following corpora test files were used: *(i)* LaPS Benchmark, which contains 60 phrases and 614 words; and *(ii)* Federal Constitution, which contains 126 phrases and 7073 words. Calculations of these metrics are performed by each library.

[5] http://jcsilva.github.io/2017/03/18/compile-kaldi-android
[6] https://gitlab.com/lucasdebatin/ocsr-android-native
[7] https://play.google.com/store/apps/details?id=com.debatin.ocsr_android_native

Each library has 10 configurations with two different sets of modifications. The two best selected for the experiments. The subsections below present the results obtained for each library.

CMUSphinx: Table 2 shows the values obtained by each configuration in the LaPS Benchmark corpus.

Table 2. Configurations of CMUSphinx Library and LaPS Benchmark Corpus

ID	WER	SER	Duration
1	49.4%	93.3%	00:03:08
2	9.6%	55%	00:03:01
3	9.1%	56.7%	00:03:17
4	11.4%	61.7%	00:05:29
5	22.1%	68.3%	00:08:21
6	38%	81.7%	00:07:53
7	8.8%	41.7%	00:04:48
8	6.7%	45%	00:05:37
9	6.2%	35%	00:09:13
10	11.9%	50%	00:11:39

Analyzing Table 2, it can be verified that the best configurations obtained have a WER lowerthan 10%. In addition, it can be seen that the best evaluation metrics come from configurations whose changed values are close to the default library configuration values. The total training duration of all settings was about 1 hour. Table 3 shows the values obtained in the Federal Constitution corpus.The results presented in Table 3 show that the tests with only one speaker have higher accuracy (WER) than the test with several speakers, presented in Table 2. Almost all configurations have a WER lower than 14%, however they have a high value in the SER metric, that is, many sentences were generated with one or more errors. The total training duration of all library settings exceeded 6 hours, and one reason for this is also the average duration of the corpus test files.

Table 3. Configurations of CMUSphinx Library and Federal Constitution Corpus

ID	WER	SER	Duration
1	14.5%	96.8%	00:22:56
2	5%	77.8%	00:22:42
3	3.2%	75.4%	00:27:17
4	4.6%	81%	00:30:29
5	5.6%	81.7%	00:32:18
6	10.8%	90.5%	00:40:43
7	3%	70.6%	00:43:36
8	2.2%	59.5%	00:56:03
9	2%	62.7%	01:17:39
10	3.4%	67.5%	01:24:58

HTK: Table 4 presents the results of the evaluation metrics obtained by the library in the LaPS Benchmark corpus.

Table 4. Configurations of HTK Library and La PS Benchmark Corpus

ID	Type	WER	SER	Duration
1	HVite (2-gram)	94.95%	100%	00:05:19
2	HVite (2-gram)	92.18%	100%	00:05:33
3	HVite (2-gram)	93.32%	100%	00:05:25
4	HVite (2-gram)	92.35%	100%	00:05:21
5	HVite (2-gram)	93.49%	100%	00:05:25
6	HVite (2-gram)	95.44%	100%	00:05:21
7	HVite (2-gram)	93.00%	100%	00:05:25
8	HVite (2-gram)	92.83%	100%	00:05:19
9	HVite (2-gram)	92.83%	100%	00:05:18
10	HVite (2-gram)	92.18%	100%	00:05:17

Comparing the results obtained by the CMUSphinx library in the same corpus (Table 2), the results of Table 4 were insignificant. In addition, when comparing the phrases of the test files with the phrases generated by the library, it turns out that all phrases have at least one error. The average training duration was 5 minutes and 22 seconds for each setting. Table 5 presents the results obtained in the Federal Constitution corpus.The values obtained in Table 5 presented better results, but worse than WER of 14%, which is one of the requirements of this work. In addition, all generated sentences have one or more errors. The average training time was 2 hours and 8 minutes for each setting.

Table 5. Configurations of HTK Library and Federal Constitution Corpus

ID	Type	WER	SER	Duration
1	HVite (2-gram)	91.98%	100%	02:08:11
2	HVite (2-gram)	89.64%	100%	02:12:03
3	HVite (2-gram)	86.17%	100%	02:09:21
4	HVite (2-gram)	86.47%	100%	02:06:09
5	HVite (2-gram)	84.97%	100%	02:06:32
6	HVite (2-gram)	93.85%	100%	01:58:51
7	HVite (2-gram)	88.60%	100%	02:03:42
8	HVite (2-gram)	83.84%	100%	02:08:38
9	HVite (2-gram)	82.47%	100%	02:11:46
10	HVite (2-gram)	84.21%	100%	02:19:31

The HTK library had the worst results. With any configuration and in both corpora, it was not possible to reach a WER below 80%. The poor results obtained by HTK are justifiable because this library is just a toolkit for building and manipulating HMM. For the recognition of continuous speech and extensive vocabulary, Julius, the Speech Recognition Engine of HTK, can be used.

Kaldi: Table 6 shows the values obtained by each Kaldi library configuration in the LaPS Benchmark corpus.

Table 6. Configurations of Kaldi Library and La PS Benchmark Corpus

ID	Type	WER	SER	Duration
1	MLP	5.05%	41.67%	00:49:17
2	RNN	2.61%	21.67%	02:24:06
3	RNN	6.19%	41.67%	14:18:37
4	RNN	8.14%	51.67%	92:54:37
5	N/A	N/A	N/A	N/A
6	tri3b	5.05%	40%	00:23:41
7	mono0a	7.17%	38.33%	00:45:13
8	tri1	6.68%	43.33%	00:42:41
9	tri1	5.37%	31.67%	00:38:46
10	tri1	6.51%	41.67%	00:40:57

Table 6 shows that the use of Artificial Neural Networks (ANN) presents the best results. However, one problem is the higher time spent in training. Therefore, the ID 5 configuration was not trained because it would take more than 100 hours. The best results were obtained using Recurrent Neural Network (RNN) and Multilayer Perceptron (MLP). In addition, classes tri3b and tri1 also showed good results. Table 7 presents the results of the library in the Federal Constitution corpus.

Table 7. Configurations of Kaldi Library and Federal Constitution Corpus

ID	Type	WER	SER	Duration
1	MLP	1.44%	46.03%	07:45:17
2	RNN	0.98%	41.27%	26:38:27
3	RNN	0.93%	38.89%	166:47:14
4	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A
6	tri3b	1.48%	46.03%	01:50:37
7	tri1	1.61%	49.21%	01:54:36
8	tri1	1.34%	38.89%	02:02:11
9	tri1	1.75%	46.03%	05:32:35
10	tri1	1.80%	48.41%	07:09:14

Examining the table 7, it can be observed that only the three trainings for the ANN were generated, since the time spent training for ID 3 exceeded 150 hours. Therefore, in this case only three configurations were selected, not four like the other libraries. The use of RNN and the tri3b and tri1 classes presented the best results of the evaluation metrics.

Desktop Performance Analysis: The tests were performed only on a desktop computer, as configured in Table 1. These tests were performed without internet connection, with no software running in parallel, and only using the corpora test files. Table 8 shows the results obtained in the LaPS Benchmark corpus. In the Figure, the "top" command, by default, displays the percentage of a single CPU, i.e. multi-core computers can have percentages higher than 100%.For each library, the best configuration was selected. The CMUSphinx and Kaldi libraries achieved the best results, but the Kaldi library stood out because it obtained: *(i)* the lowest WER percentage; *(ii)* the lowest average processor usage; and *(iii)* the lowest average memory usage.

Table 8. Performance of Libraries in LaPS Benchmark Corpus

Library	ID	Duration	xRT	Processor (average)	Memory(average)
CMUSphinx	2	00:00:11	0.040	97.50%	0.50%
CMUSphinx	3	00:00:19	0.069	98.09%	0.50%
CMUSphinx	8	00:00:36	0.130	98.69%	0.50%
CMUSphinx	9	00:01:01	0.221	99.20%	0.50%
HTK	2	00:02:44	0.594	99.17%	0.60%
HTK	4	00:02:43	0.590	98.68%	0.60%
HTK	9	00:02:44	0.594	99.00%	0.60%
HTK	10	00:02:41	0.583	98.79%	0.60%
Kaldi	1	00:02:18	0.500	98.26%	0.43%
Kaldi	2	00:03:31	0.764	97.79%	0.39%
Kaldi	6	00:00:14	0.050	91.92%	0.20%
Kaldi	9	00:00:19	0.069	96.73%	0.21%

Table 9 presents the performance results obtained by using the Federal Constitution corpus. In order to evaluate the results in this corpus, the best configuration of each library was selected. It can be seen that the CMUSphinx and Kaldi libraries achieved the best results. In this corpus, the Kaldi library also stood out, obtaining: *(i)* the lowest percentage WER; *(ii)* the smallest value xRT; *(iii)* the lowest average processor usage; and *(iv)* the lowest average memory usage.

Table 9. Performance of Libraries in Federal Constitution Corpus

Library	ID	Duration	xRT	Processor (average)	Memory (average)
CMUSphinx	3	00:02:29	0.047	99.27%	0.50%
CMUSphinx	4	00:04:06	0.077	98.35%	0.50%
CMUSphinx	8	00:04:29	0.084	99.07%	0.50%
CMUSphinx	9	00:07:45	0.146	98.92%	0.50%
HTK	3	01:26:02	1.620	99.14%	0.70%
HTK	5	01:25:52	1.617	99.20%	0.70%
HTK	8	01:25:58	1.619	99.10%	0.70%
HTK	9	01:26:07	1.622	98.52%	0.70%
Kaldi	3	02:39:45	3.008	98.11%	0.95%
Kaldi	6	00:01:30	0.028	94.76%	0.20%
Kaldi	8	00:01:54	0.036	97.50%	0.37%

According to Tables 8 and 9, the libraries demanded more processing resources than RAM. In addition, the use of ANN presented the best results (2.61% in the LaPS Benchmark corpus and 0.93% in the Brazilian Constitution corpus). On the other hand, it requires a great computational cost, in some cases exceeding two hours of processing, thus compromising its use in mobile devices. Still according to Table 8 and Table 9, the best results were obtained from the Kaldi library. However, the implementation of this library on mobile devices was unsuccessful. Therefore, the CMUSphinx library, which have results very similar to those of the Kaldi library, was chosen.

PERFORMANCE ANALYSIS ON MOBILE DEVICES

For the tests on mobile devices the CMUSphinx library was used. The tests were performed on 11 devices, with different versions of Android (upper 4.1) and hardware configurations. In each device it was requested to: *(i)* disable any form of internet connection (wi-fi, mobile data); *(ii)* remove the cell from the power supply, if there was any; *(iii)* verify that the device had at least 30% battery; and *(iv)* close all open applications. All tested mobile devices have a low initial processor usage and sufficient memory to perform the tests. Table 10 shows the configurations of the mobile devices used in the tests.

Table 10. Configurations of Mobile Devices

Device	Android	Hardware Configuration
Samsung SM-T110	4.4.2	Processor: Dual-Core of 1.2 GHz RAM Memory: 1 GB
Samsung SM-J200BT	5.1.1	Processor: Quad-Core of 1.3 GHz RAM Memory: 1 GB
Quantum Fly	6.0	Processor: Deca-Core of 2.1 GHz RAM Memory: 3 GB
Samsung SM-J500M	6.0.1	Processor: Quad-Core of 1.2 GHz RAM Memory: 1.5 GB
LGE LG-M700	7.1.1	Processor: Octa-Core of 1.4 GHz RAM Memory: 2 GB
Xiaomi Redmi 4X	7.1.2	Processor: Quad-Core of 1.4 GHz RAM Memory: 3 GB
Motorola Moto Z (2)	8.0.0	Processor: Octa-Core of 2.35 GHz RAM Memory: 6 GB
Motorola XT1635-02	8.0.0	Processor: Octa-Core of 2 GHz RAM Memory: 3 GB
Motorola Moto G (5S)	8.1.0	Processor: Octa-Core of 1.4 GHz RAM Memory: 2 GB
Samsung SM-J610G	8.1.0	Processor: Quad-Core of 1.4 GHz RAM Memory: 3 GB
Xiaomi Mi A2	9.0.0	Processor: Octa-Core of 2.2 GHz RAM Memory: 4 GB

Table 11[8] shows the results obtained during the execution of the LaPS Benchmark corpus on mobile devices. On all devices, it is possible to notice: *(i)* the average memory usage was below 70 MB; and *(ii)* the average processor usage was below 50%. It is important to note that the processing of the test files by the library does not consume the device's battery resources. Table 12[9] presents the results obtained during the execution of the Federal Constitution corpus. According to Table 12, it can also be seen that in all devices: *(i)* the average memory usage was below 80 MB; *(ii)* the average processor usage was below 50%; and *(iii)* the processing of the test files did not consume the battery resources of the devices. It is also important to show the behavior of the variable xRT, which computes the processing time of the audio file. Table 13 shows a comparison between the processor usage and the xRT value obtained for each corpus. This table was developed based on the hardware configurations of the devices (Table 10) and the xRT values of Tables 11 and 12. Based on Table 13, it can be seen that the xRT values are low in almost all processors, that is, virtually all of them have obtained values below 1. However, the time for processing each audio of the test corpora in a Dual-Core 1.2 GHz processor was high because this processor is old and has only two cores with only 1.2 GHz each. The values of xRT obtained with an Octa-Core 2.2 GHz processor were similar to the values obtained in the analysis of performance in the desktop computer.

DISCUSSION

The tool implemented in this work presents two versions: desktop and mobile. The main goal of this tool was to find the best off-line solution for continuous speech recognition. The desktop version was useful to find the library that had the best performance results in the tests performed on a desktop computer. In this version, the xRT value was calculated and information about processor and memory usage was collected. The Kaldi library achieved the best results, but it was not possible to implement it on mobile devices due to permission restrictions. For this reason, the CMUSphinx library was used, which obtained results similar to Kaldi. The HTK library presented the worst results, with the best WER value obtained being higher than 80%. The results obtained in the analysis of the performance of the desktop version demonstrated that the use of ANN in the continuous voice recognition requires a high computational cost but presented the best results. The best cost-benefit ratio between accuracy and performance was obtained by applying HMM. The tests were performed on several mobile devices. In this way, it can be observed that in all tested versions of Android, the HMM library has successfully performed continuous offline speech recognition. It is important to note that the latest versions of Android have processed the test files in real time. This was possible due to the modern hardware configurations of the mobile devices. Experiments were carried out with two speech corpora, one with several speakers and the other with only one speaker. The corpus with only one speaker presented the best results for the WER rate. Thus, it can be concluded that the presence of several speakers is a complexity factor of continuous speech recognition.

Finally, Table 14 presents the comparison between the selected works. They encompass the state of the art of the research problem, with the best solution found in that article. Only the work related to identifiers (Pakoci et al., 2017, 2018) presented error rate results in offline mode for mobile devices. Our solution uses the main acoustic and language techniques for the implementation of the solution, as it can be seen in Table 14. This work is the state-of-the-art for Brazilian Portuguese language, since there are no works with the same objective for this language. It is also possible to remark the average WER obtained in the articles as the state-of-the-art is 14.01% for several languages. In this work, the WER obtained for Brazilian Portuguese was better than this average, both for the corpus with one speaker (3.2%) and for the corpus with several speakers (9.6%). Some of the limitations presented by Pakoci et al. (2017, 2018) are the lack of comparison metrics for the optimization of processor, memory and battery usage in mobile devices. These metrics, presented in our approach, are among those that can be considered very relevant to the context of the solution. Moreover, these works present the use of DNN (MLP and RNN) implemented through the Kaldi library, which require a high computational cost. In both works two voice corpora are used: *(i)* 154 hours, with 87 thousand statements, with an average of 15 words each, with 21 male announcers and 27 female announcers; and *(ii)* 61 hours, with 170 male and 181 female speakers. With a corpus of text containing about 1.5 million words. We can also highlight the work from Abushariah (2018) which presented results in the online mode with WER of 2.68%. It used the same techniques and the same library of our work, although it was used for an online solution. In addition, there was a significant difference in the size of the corpus speech for training. While we had limitations on corpora size, Abushariah (2018)presented a solution based on training in a much larger corpus, composed of 41,005 sentences. The authoranalysedspeech data collected from 36 native speakers from 11 different Arab countries, summing about 45 hours of speech. It is also worthwhileto mention Georgescu et al. (2017), which presented aDNN, from Kaldi library, as a solution for the WER reduction. The author did not use data in offline mode and on mobile devices. Three corpora were used: *(i)* 100 hours in a quiet environment; *(ii)* 28 hours of talk shows (affected by noise) and news (clean speech); and *(iii)* 103 hours of talk time. Two corpora of texts, including 315 million words collected from news sites and 40 million words of meetings transcripts, were analysed.

CONCLUSIONS AND FUTURE WORK

This research had as its main objective the development of offline continuous speech recognition for mobile devices in the Brazilian Portuguese language. Also, checking the performance analysis on various mobile devices with the Android operating system. Performance analysis has demonstrated that it is possible to run the CMUSphinx library in applications for real-time speech recognition.

[8]All test files of the corpus were run, with duration of 00:04:36 (276 seconds).
[9]In section 3 it was described that for this corpus only 30 test files were used, with duration of 00:12:32 (752 seconds), thus changing the calculation of the factor xRT.

Table 10. Configurations of Mobile Devices

Device	Android	Hardware Configuration
Samsung SM-T110	4.4.2	Processor: Dual-Core of 1.2 GHz RAM Memory: 1 GB
Samsung SM-J200BT	5.1.1	Processor: Quad-Core of 1.3 GHz RAM Memory: 1 GB
Quantum Fly	6.0	Processor: Deca-Core of 2.1 GHz RAM Memory: 3 GB
Samsung SM-J500M	6.0.1	Processor: Quad-Core of 1.2 GHz RAM Memory: 1.5 GB
LGE LG-M700	7.1.1	Processor: Octa-Core of 1.4 GHz RAM Memory: 2 GB
Xiaomi Redmi 4X	7.1.2	Processor: Quad-Core of 1.4 GHz RAM Memory: 3 GB
Motorola Moto Z (2)	8.0.0	Processor: Octa-Core of 2.35 GHz RAM Memory: 6 GB
Motorola XT1635-02	8.0.0	Processor: Octa-Core of 2 GHz RAM Memory: 3 GB
Motorola Moto G (5S)	8.1.0	Processor: Octa-Core of 1.4 GHz RAM Memory: 2 GB
Samsung SM-J610G	8.1.0	Processor: Quad-Core of 1.4 GHz RAM Memory: 3 GB
Xiaomi Mi A2	9.0.0	Processor: Octa-Core of 2.2 GHz RAM Memory: 4 GB

Table 11. Performance of LaPS Benchmark Corpus

Device	Duration	xRT	Battery	Processor (average)	Memory (average)
Samsung SM-T110	01:53:27	24.6	0%	32%	17.49 MB
Samsung SM-J200BT	00:01:11	0.257	0%	25%	27.22 MB
Quantum Fly	00:00:41	0.149	0%	45%	67.32 MB
Samsung SM-J500M	00:00:46	0.167	1%	25%	41.45 MB
LGE LG-M700	00:00:36	0.130	0%	21%	63.05 MB
Xiaomi Redmi 4X	00:00:34	0.123	0%	25%	50.43 MB
Motorola Moto Z (2)	00:00:28	0.101	0%	6%	21.98 MB
Motorola XT1635-02	00:00:27	0.978	0%	1%	53.39 MB
Motorola Moto G (5S)	00:00:43	0.156	0%	27%	43.81 MB
Samsung SM-J610G	00:00:43	0,156	0%	3%	30,96 MB
Xiaomi Mi A2	00:00:10	0,036	0%	12%	27,30 MB

Table 12. Performance of Federal Constitution Corpus

Device	Duration	xRT	Battery	Processor(average)	Memory(average)
Samsung SM-T110	02:44:02	13.08	1%	33%	21.52 MB
Samsung SM-J200BT	00:04:51	0.387	1%	20%	32.97 MB
Quantum Fly	00:02:55	0.233	1%	46%	74.97 MB
Samsung SM-J500M	00:05:05	0.406	0%	24%	38.31 MB
LGE LG-M700	00:02:11	0.174	1%	36%	59.80 MB
Xiaomi Redmi 4X	00:04:00	0.319	1%	25%	51.55 MB
Motorola Moto Z (2)	00:00:28	0.037	0%	22%	26.67 MB
Motorola XT1635-02	00:01:36	0.128	1%	9%	55.47 MB
Motorola Moto G (5S)	00:01:58	0.157	1%	28%	53.04 MB
Samsung SM-J610G	00:02:17	0.182	0%	20%	33.28 MB
Xiaomi Mi A2	00:00:32	0.043	0%	4%	30.51 MB

Table 13. Comparative Between Processor and xRT Value

Processor	xRTLaPS Benchmark Corpus	xRT Federal Constitution Corpus
Dual-Core of 1.2 GHz	24.66	13.08
Quad-Core of 1.2 GHz	0.167	0.406
Quad-Core of 1.3 GHz	0.257	0.387
Quad-Core of 1.4 GHz	0.123	0.319
Quad-Core of 1.4 GHz	0.156	0.182
Octa-Core of 1.4 GHz	0.130	0.174
Octa-Core of 1.4 GHz	0.156	0.157
Octa-Core of 2 GHz	0.978	0.128
Octa-Core of 2.2 GHz	0.036	0.043
Octa-Core of 2.35 GHz	0.101	0.037
Deca-Core of 2.1 GHz	0.149	0.233

Table 14. Comparative Between the Related Works and This Paper

Reference	Acoustic Model	Language Model	Library	Idiom	WER
Abushariah (2018)	HMM	1, 2 and 3-gram	CMU Sphinx	Arabic (11 countries)	2.68%
Georgescu et al. (2017)	RNN and HMM	1, 2 and 3-gram	Kaldi	Romanian	4.5%
Kipyatkova and Karpov (2017)	HMM	RNN-LM and 3-gram	HTK	Russian	22.87%
Laleyeet al. (2016)	"monofone" and "trifone" from Kaldi's library	3-gram (SRILM tool)	Kaldi	Fongbe	14.83%
Naing et al. (2015)	MLP and HMM	Word-base	-	Myanmar	15.63%
Pakoci et al. (2017)	MLP and HMM	1, 2 and 3-gram	Kaldi	Serbian	12.01%
Pakoci et al. (2018)	MLP, RNN and HMM	1, 2 and 3-gram	Kaldi	Serbian	7.23%
Phull and Kumar (2016)	HMM and HMM	2 and 3-gram	CMU Sphinx	English Indian	19%
Tachbelie et al. (2014)	HMM	3-gram (SRILM tool)	CMU Sphinx	Amharic	13.3%
Zhang et al. (2015)	MLP and HMM	2 and 3-gram	-	Mongol	12.37%
This paper	HMM	1, 2 and 3-gram	CMU Sphinx	Brazilian Portuguese	3.2%

However, this can only be done in devices with hardware configurations with higher computational resources, since they presented a low score in xRT. For devices with hardware configurations with limited computational resources, it is only possible to use this library in applications that perform interview audio transcriptions, for example. All the tests were performed without internet connection, seeking for emphasizing that this continuous speech recognition can work successfully in environments without internet. The tests corroborated the feasibility of applyingspeech recognition of the Brazilian Portuguese language in offline mobile devices. The main contributions of this work are: *(i)* the implementation of an off-line speech recognition solution for the Brazilian Portuguese language; *(ii)* the development of a tool for performance analysis of the CMUSphinx, HTK and Kaldi libraries on a desktop computer; and *(iii)* the development of a tool for analyzing the performance of the CMUSphinx library on various mobile devices running the Android operating system. It is worth mentioning that all tools developed in this work are available free of charge in GitLab, serving as a basis for future work.

In addition, it is also possible to highlight as a contribution of the work the comparison carried out between the training configurations for each library in order to locate the best evaluation metric. This speech recognition implementation can be used in software and applications: *(i)* that assist in the communication of people with disabilities; *(ii)* companies that streamline the work of employees; and *(iii)* that require this function in areas without connection to the Internet. The performance analysis tool can be used to measure the required performance of a new corpus speech. Throughout the development of this work, some possibilities of improvement and continuation could be identified from future research, which includes:

- Implementing the Kaldi library on mobile devices without the need for "superuser" permission, so it will be necessary to compile the library files in a different way;
- Carrying out a study on how to reduce the computational cost (processing) required by ANN that perform continuous speech recognition to be applied on mobile devices. For this, it will be necessary to find and implement ANN that performs better;
- Using intelligent software agents to get the training settings with the best WER rate. For this, it will be necessary to perform the implementation of this agent and configure it to change the settings of the libraries;
- Analyzing the performance of offline speech recognition on mobile devices with the IOS operating system. For this, it will be possible to use the same library used in Android.

ACKNOWLEDGMENT

We thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for financial support to carry out this research work.

REFERENCES

Abushariah, MA 2018. TAMEEM V1.0: speakers and text independent Arabic automatic continuous speech recognizer.International Journal of Speech Technology, 20(2):261-280.

Bauer MW, Aarts B (2000). Corpus construction: a principle for qualitative data collection. In: Martin W. Bauer and George Gaskell (eds.) Qualitative researching: with text, image and sound. London, UK: Sage. pp. 19-37.

Coppin B (2004) Artificial Intelligence Illuminated.Sudbury, USA: Jones & Bartlett Learning.

DebatinL, Haendchen Filho A, Dazzi, RLS (2018). Offline Speech Recognition Development: A Systematic Review of the Literature. Proceedings of the International Conference on Information Systems, San Francisco, USA, pp. 551-558.

FalaBrasil (2019). Grupo Falabrasil - UFPa. [Online]. http://www.laps.ufpa.br/falabrasil

Ferreira MVG, Souza JF (2017). Use of Automatic Speech Recognition Systems for Multimedia Applications. Proceedings of 23rd Brazillian Symposium on Multimedia and the Web (Webmedia),Gramado, Brazil, ACM, pp. 139-176.

Georgescu A, Cucu H, Burileanu C (2017). SpeeD's DNN approach to Romanian speech recognition.Proceedinsgs of the International Conference on Speech Technology and Human-Computer Dialogue (SpeD), Bucharest, Romania, IEEE, pp. 1-8.https://doi.org/10.1109/SPED.2017.7990443.

Georgescu AL, Pappalardo A, Cucu H, Blott M (2021). Performance vs. hardware requirements in state-of-the-art automatic speech recognition. Journal of Audio Speech Music Processing,28. https://doi.org/10.1186/s13636-021-00217-4

Grossman RL (2009). The Case for Cloud Computing.IT Professional, 11(2):23-27.

Gupta K, OwensJD (*2011*). Compute & memory optimizations for high-quality speech recognition on low-end GPU processors.Proceedings of the *18thInternational Conference on High Performance Computing*, pp. 1-10, https://doi.org/10.1109/HiPC.2011.6152741.

Hearst, MA (2011). 'Natural' search user interfaces.Communications of the ACM, 54(11):60-67.

Huang X, DengL (2010). An overview of modern speech recognition.In Handbook of Natural Language Processing, 2nded. London, UK: Chapman and Hall/CRC, pp. 339-366.

Jurafsky D, MartinJH (2008). Speech and Language Processing: An Introduction to Natural Language Processing, in Computational Linguistics, and Speech Recognition, 2nded. Upper Saddle River, USA: Prentice-Hall.

Kipyatkova IS, KarpovAA (2017). A study of neural network Russian language models for automatic continuous speech recognition systems.Automation and Remote Control, 78(5):858-867.

Laleye FAA, Besacier L, Ezin EC, Motamed C (2016). First automatic fongbe continuous speech recognition system: Development of acoustic models and language models. Proceedings of the Federated Conference on Computer Science and Information Systems, Gdansk, Poland, pp. 477-482.

LeeK, Hon H, Reddy R (1990). An Overview of the SPHINX Speech Recognition System, IEEE Transactions on Acoustic Speech, and Signal Processing, 38(1):35-45.

Meyer J, Dentel L, Meunier F (2014). Speech Recognition in Natural Background Noise. *PLOS ONE*, 9(1). https://doi.org/10.1371/annotation/012d9419-8135-40ab-8c81-ce46e8e708d0

Naing HMS, Hlaing AM, Pa WP, Hu X, Thu YK, Hori C, Kawai H (2015). A Myanmar large vocabulary continuous speech recognition system.Proceedings of the Asia-Pacific Signal and Information Processing Association, Hong Kong, pp. 320-327.

National Academies of Sciences, Engineering, and Medicine (2017). Augmentative and Alternative Communication and Voice Products and Technologies. In: The Promise of Assistive Technology to Enhance Activity and Work Participation. Washington, DC, USA: The National Academies Press. https://doi.org/10.17226/24740.

Neto N, Patrick C, Klautau A, Trancoso I (2011). Free tools and resources for Brazilian Portuguese speech recognition. Journal of the Brazilian Computing Society, 17:53–68. https://doi.org/10.1007/s13173-010-0023-1

Pakoci E, Popovic B, Pekar D (2017). Language model optimization for a deep neural network based speech recognition system for Serbian.Proceedings of the International Conference on Speech and Computer, Hatfield, UK, pp. 483-492.

Pakoci E, Popovic B, Pekar D (2018). Improvements in Serbian Speech Recognition Using Sequence-Trained Deep Neural Networks. Artificial Intelligence, Knowledge and Data Engineering, 1:53-76.

Phull DK, KumarGB (2016). Investigation of Indian English Speech Recognition using CMU Sphinx.International Journal of Applied Engineering Research, 11(6):4167-4174.

Povey D, Ghoshal A, Boulianne G, Burget L, Glembek O, Goel N, Hannemann M, Motlıcek P, Qian Y, Schwarz P, SilovskyJ, Stemmer G, Vesely K (2011). The Kaldi Speech Recognition Toolkit.Proceedins of the IEEE Workshop on Automatic Speech Recognition and Understanding, Big Island, USA, IEEE, pp. 1-4.

SerizelR, BisotV, EssidS, RichardG (2017). Acoustic Features for Environmental SoundAnalysis.In: T. Virtanen; M. D. Plumbley; D. Ellis. Computational Analysis of Sound Scenesand Events, Springer, pp.71-101.

Silva E, Baptista L, Fernandes H,Klauta A (2005). Desenvolvimento de um Sistema de Reconhecimento Automático de Voz Contínua com Grande Vocabulário para o Português Brasileiro.ProceedingsoftheWorkshop em Tecnologia da Informação e Linguagem Humana, São Leopoldo, Brazil, pp. 2258-2267.

Soni VD (2019). Speech Recognition: Transcription and transformation of human speech. International Journal on Integrated Education, 2(5):257-262.

Tachbelie MY, Abate ST, Besacier L (2014). Using different acoustic, lexical and language modeling units for ASR of an under-resourced language – Amharic.Speech Communication 56(1):181-194.

Tevah RT (2006). Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro.MScdissertation, COPPE, UFRJ, Rio de Janeiro, Brazil.

Yu D, Deng L (2015). Automatic Speech Recognition: A Deep Learning Approach. London, UK: Springer.

Zhang H, Bao F, Gao G (2015). Mongolian speech recognition based on deep neural networks.Proceedings of the 14thChinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data,Guangzhou, China, pp.180-188.

****