Chapter 13

VOICECONET: A Collaborative Framework for Speech-Based Computer Accessibility with a Case Study for Brazilian Portuguese

Nelson Neto, Pedro Batista and Aldebaro Klautau

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/47835

1. Introduction

In recent years, the performance of personal computers has evolved with the production of ever faster processors, a fact that enables the adoption of speech processing in computer-assisted education. There are several speech technologies that are effective in education, among which text-to-speech (TTS) and automatic speech recognition (ASR) are the most prominent. TTS systems [45] are software modules that convert natural language text into synthesized speech. ASR [18] can be seen as the TTS inverse process, in which the digitized speech signal, captured for example via a microphone, is converted into text.

There is a large body of work on using ASR and TTS in educational tasks [14, 37]. All these speech-enabled applications rely on *engines*, which are the software modules that execute ASR or TTS. This work proposes a collaborative framework and associated techniques for constructing speech engines and adopts accessibility as the major application. The network has an important social impact in decreasing the recent digital divide among speakers of commercially attractive and underrepresented languages.

The incorporation of computer technology in learning generated multimedia systems that provide powerful "training tools", explored in computer-assisted learning [34]. Also, Web-based learning has become an important teaching and learning media [52]. However, the financial cost of both computer and software is one of the main obstacle for computer-based learning, especially in developing countries like Brazil [16].

The situation is further complicated when it comes to people with special needs, including visual, auditory, physical, speech, cognitive, and neurological disabilities. They encounter serious difficulties in having access to this technology and hence to knowledge. For example, according to the Brazilian Institute of Geography and Statistics (IBGE), 14.5% of the Brazilian

population has some type of disability as detailed in Table 1. It is important then to understand how speech technologies can help educating people with disabilities [37].

Disability	Description	Number of people
Visual	including blindness, low vision, and reduced color perception	16,644,842
Auditory	including total or parcial hearing impairments	5,735,099
Physical	motor disabilities can include weakness, limitations of muscular control (such as involuntary movements, lack of coordination, or paralysis), limitations of sensation, joint problems, or missing limbs	9,355,844
Mental	including cognitive and neurological disabilities	2,844,936

Table 1. Profile of Brazilian people with disabilities based on data provided by IBG [20] (the total population is 190,732,694).

For the effective use of speech-enabled applications in education and assistive systems, reasonably good engines must be available [30, 43]. Besides, the cost of softwares and equipments cannot be prohibitive.

This work presents some results of an ambitious project, which aims at using the Internet as a collaborative network and help the academy and software industry in the development of speech science and technology for any language, including Brazilian Portuguese (BP). The goal is to collect, develop, and deploy resources and softwares for speech processing using a collaborative framework called VOICECONET. The public data and scripts (or software *recipes*) allow to establish baseline systems and reproduce results across different sites [48]. The final products of this research are a large-vocabulary continuous speech recognition (LVCSR) system and a TTS system for BP.

The remainder of the chapter is organized as follows. Section 2 presents a description of ASR and TTS systems. Section 3 describes the proposed collaborative framework, which aims at easing the task of developing ASR and TTS engines to any language. Section 4 presents the developed resources for BP such as speech databases and phonetic dictionary. The baseline results are presented in Section 5. Finally, Section 6 summarizes our conclusions and addresses future works.

2. Background on speech recognition and synthesis

First, this section provides a brief introduction to ASR and TTS systems. The last two topics present evaluation methods and development tools.

2.1. Automatic speech recognition (ASR)

The typical ASR system adopts a statistical approach based on *hidden Markov models* (HMMs) [22], and is composed by five main blocks: front end, phonetic dictionary, acoustic model, language model and decoder, as indicated in Figure 1. The two main ASR applications are *command and control* and *dictation* [18]. The former is relatively simpler, because the

language model is composed by a grammar that restricts the acceptable sequences of words. The latter typically supports a vocabulary of more than 60 thousand words and demands more computation.

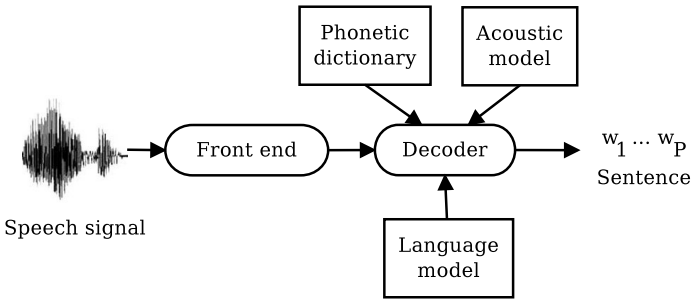

Figure 1. The main constituent blocks of a typical ASR system.

The conventional front end extracts segments (or *frames*) from the speech signal and converts, at a constant *frame rate* (typically, 100 Hz), each segment to a vector $\mathbf{x}$ of dimension L (typically, $L = 39$). It is assumed here that T frames are organized into a $L \times T$ matrix $\mathbf{X}$, which represents a complete sentence. There are several alternatives to parameterize the speech waveforms. In spite of the mel-frequency cepstral coefficients (MFCCs) analysis being relatively old [10], it has been proven to be effective and is used pervasively as the input to the ASR back end [18].

The language model of a dictation system provides the probability $p(\mathcal{T})$ of observing a sentence $\mathcal{T} = [w_1, \ldots, w_P]$ of P words. Conceptually, the decoder aims at finding the sentence $\mathcal{T}^*$ that maximizes a posterior probability as given by

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} p(\mathcal{T}|\mathbf{X}) = \arg\max_{\mathcal{T}} \frac{p(\mathbf{X}|\mathcal{T})p(\mathcal{T})}{p(\mathbf{X})}, \tag{1}$$

where $p(\mathbf{X}|\mathcal{T})$ is given by the acoustic model. Because $p(\mathbf{X})$ does not depend on $\mathcal{T}$, the previous equation is equivalent to

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} p(\mathbf{X}|\mathcal{T})p(\mathcal{T}). \tag{2}$$

In practice, an empirical constant is used to weight the language model probability $p(\mathcal{T})$ before combining it with the acoustic model probability $p(\mathbf{X}|\mathcal{T})$.

Due to the large number of possible sentences, Equation (2) cannot be calculated independently for each candidate sentence. Therefore, ASR systems use data structures such as lexical trees that are hierarchical, breaking sentences into words, and words into *basic units* as phones or triphones [18].

A phonetic dictionary (also known as lexical model) provides the mapping from words to basic units and vice-versa. For improved performance, continuous HMMs are adopted, where the output distribution of each state is modeled by a mixture of Gaussians, as depicted in Figure 2. The typical HMM topology is "left-right", in which the only valid transitions are staying at the same state and moving to the next.

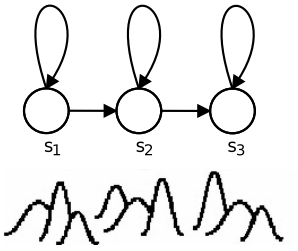

Figure 2. Pictorial representation of a left-right continuous HMM with three states $s_i, i = 1, 2, 3$ and a mixture of three Gaussians per state.

To reduce the computational cost of searching for $\mathcal{T}^*$ (decoding), hypotheses are pruned, i.e., some sentences are discarded and Equation (2) is not calculated for them [11]. In summary, after having all models trained, an ASR at the test stage uses the front end to convert the input signal to parameters and the decoder to search for the best sentence $\mathcal{T}$.

The acoustic and language models can be fixed during the test stage but adapting one or both can lead to improved performance. For example, the topic can be estimated and a specific language model used. This is crucial for applications with a technical vocabulary such as X-ray reporting by physicians [2]. The adaptation of the acoustic model is also important [25].

The ASR systems that use speaker independent models are convenient but must be able to recognize with a good accuracy any speaker. At the expense of requesting the user to read aloud some sentences, speaker adaptation techniques can tune the HMM models to the target speaker. The adaptation techniques can also be used to perform environmental compensation by reducing the mismatch due to channel or additive noise effects.

In this work, an ASR engine is considered to be composed by the decoder and all the required resources for its execution (language model, etc.). Similarly, a TTS engine consists of all software modules and associated resources, which will be briefly discussed in the sequel.

2.2. Text-to-speech (TTS)

A typical TTS system is composed by two parts: the front end and the back end [9]. Figure 3 depicts a simple functional diagram of a TTS system.

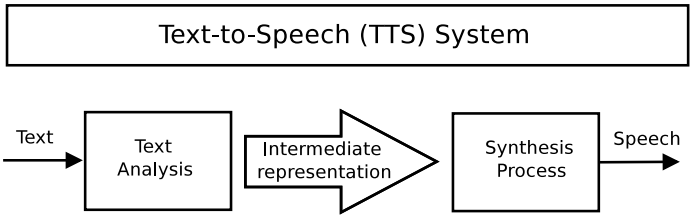

Figure 3. Functional diagram of a TTS system showing the front and back ends, responsible by the text analysis and speech synthesis, respectively.

The front end is language dependent and performs text analysis to output information coded in a way that is convenient to the back end. For example, the front end performs

text normalization, converting text containing symbols like numbers and abbreviations into the equivalent written-out words. It also implements grapheme-to-phone conversion, syllabification and syllable stress determination, which assign phonetic transcriptions to each word and marks the text with prosodic information. Phonetic transcriptions and prosody information together compose the (intermediate) symbolic linguistic representation that is output by the front end.

The back end is typically language independent and includes the synthesizer, which is the block that effectively generates sound. With respect to the technique adopted for the back end, the main categories are the formant-based, concatenative and, more recently, HMM-based [45]. Historically, TTS systems evolved from a knowledge-based paradigm to a pragmatic data-driven approach.

2.3. Evaluation metrics

In most ASR applications the figure of merit of an ASR system is the word error rate (WER). In this work, the WER is defined as

$$\text{WER} = \frac{D + R}{W} \times 100\%, \tag{3}$$

where W is the number of words in the input sequence, R and D are the number of replacement and deletion errors on the recognized word sequence, respectively, when compared with the correct transcription.

Another metric for evaluating an ASR system is the real-time factor (xRT). The xRT is obtained by dividing the time that the system spends to recognize a sentence by its time duration. A lower xRT indicates a faster recognition.

The most common metric for evaluating a language model is the probability $p(\mathbf{T})$ that the model assigns to some test data $\mathbf{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_S\}$ composed of S sentences. Independence among the sentences is assumed, which leads to $p(\mathbf{T}) = p(\mathcal{T}_1)p(\mathcal{T}_2)\ldots p(\mathcal{T}_S)$. Two measures are derived from this probability: perplexity and cross-entropy [18]. The cross-entropy $H_p(\mathbf{T})$ is defined as

$$H_p(\mathbf{T}) = -\frac{1}{W_{\mathbf{T}}} \log_2 p(\mathbf{T}), \tag{4}$$

where $W_{\mathbf{T}}$ is the number of words in $\mathbf{T}$.

The perplexity (PP) is the inverse of the average conditional probability of a next word, and is related to the cross-entropy $H_p(\mathbf{T})$ by

$$\text{PP} = 2^{H_p(\mathbf{T})}. \tag{5}$$

Lower cross-entropies and perplexities indicate less uncertainty in predicting the next word and, for a given task (vocabulary size, etc.), typically indicate a better language model.

With respect to TTS, the quality of speech outputs can be measured via two factors: intelligibility and pleasantness. Intelligibility can be divided into segmental intelligibility, which indicates how accurately spoken sentences have been received by the user, and

comprehension, which measures how well the spoken sentences are understood. Segmental intelligibility can be measured in a similar way to WER by comparing transcriptions and reference messages. Comprehension can be measured by using questions or tasks which require listeners to understand the meaning of the messages.

Pleasantness of speech can be measured by collecting a large number of user opinions and using, for example, the mean opinion score (MOS) protocol [27]. The MOS is generated by averaging the results of a set of subjective tests where a number of listeners rate the heard audio quality of sentences read aloud by TTS softwares. It should be noted that intelligibility and pleasantness are related but not directly correlated. They also have a significant effect on user's acceptance: unpleasant speech output can lead to poor satisfaction with an otherwise sophisticated system.

2.4. Engines and tools for writing speech-enabled applications

The starting point to deploy a speech-enabled application is to count with an engine (ASR, TTS or both). There are commercial solutions from Microsoft [28], Nuance [29], and other companies. Currently, not all languages are supported, especially the so-called underrepresented ones. Therefore, a key aspect is to have available engines for the target language.

With respect to free engines, the MBROLA project [13] offers a set of TTS engines for 34 languages, including BP. Julius [24] and Sphinx-4 [51] are examples of open source decoders that can be used in association with the required resources to build ASR engines. For embedded (e.g., smartphones) applications, Julius and PocketSphinx [19] are the most popular engines. Research groups have made available engines for English, Japanese and other languages based on open source decoders Silva et al. [40].

Having an engine available, an *application programming interface* (API) eases the task of developing softwares for the final users, i.e., speech-enabled applications. The APIs specify a cross-platform interface to support command and control recognizers, dictation systems and speech synthesizers. As such, they contain not only the required TTS and ASR functionality but also numerous methods and events that allow programmers to query the characteristics of the underlying engine. An engine typically has its own API but there are at least two APIs that were designed for general use: the speech API from Microsoft [33] and the Java Speech API [21].

It should be noted the APIs for user interface accessibility, such as the Microsoft Active Accessibility platform and the AT-SPI toolkit on Linux. These softwares are designed to help developing assistive technology products, like screen readers and keyboards, interact with standard and custom user interface elements of an application (or the operating system), in order to provide better access for individuals who have physical or cognitive difficulties, impairments, or disabilities.

2.5. Tools for developing engines

Besides the research interest, two reasons for investing on the development of an engine is the lack of support to a given language and the need to minimize cost.

There are some free and mature software tools for building speech recognition and synthesis engines. HTK [55] is the most widespread toolkit used to build and adapt acoustic models, as well as HTS [54] (modified version of HTK) for building speech synthesis system. Statistical language models can be created with SRILM [44] or HTK.

Festival [4] offers a general multi-lingual (currently English and Spanish) framework for building speech synthesis systems. Another open source framework for TTS is the MARY platform [36]. Currently, MARY supports the German, English and Tibetan languages.

3. A collaborative framework for improved ASR and TTS

Having briefly discussed how the industry and academy are positioned to promote the development and adoption of speech-enabled applications, this section focuses on a key aspect for adopting this technology in education: using a collaborative network to increase the availability of engines.

3.1. The challenge of developing speech-enabled educational softwares

As mentioned, minimizing the cost of the application deployment can be the main reason to use a free engine or develop one. But, initially, this work has been motivated by circumventing the lack of an ASR engine for BP. For instance, when this work originated, the only desktop software for ASR dictation in BP was the IBM ViaVoice, which had been discontinued. Therefore, the intention of creating software for educating physically-impaired people was unfeasible unless engines were developed.

However, the task of designing a speech engine is not trivial and the existence of a collaborative environment, as proposed in this work, can be a decisive aspect. The availability of corpora ("databases") is a conditioning factor for the development of engines, and while there are many public resources for some languages (e.g., English and Japanese), the resources for underrepresented languages are still limited. A typical corpus for training and testing speech recognition and synthesis systems has speech files with their associated transcriptions, large amount of written texts and a phonetic dictionary.

The phonetic dictionary is an essential building block that does the correspondence between the orthography and the pronunciation(s). In practice, building a pronunciation dictionary for ASR is very similar to developing a grapheme-to-phone (G2P) module for TTS systems. In fact, a dictionary can be constructed by invoking a pre-existent G2P module. The task of designing a G2P module is difficult to perform and several techniques have been adopted over the last decade [6, 46].

As discussed, speech recognition and synthesis are data-driven technologies that require a relatively large amount of labeled data in order to achieve acceptable accuracy rates. The state-of-art HMM-based ASR and TTS methods also require a database, with multiple speakers in the case of ASR. In response to these needs, the next section suggests using the Internet as a collaborative network.

3.2. The proposed VOICECONET platform

The proposed collaborative framework aims at collecting desktop speech and text data at negligible costs for any language, in order to provide larger training data for ASR and TTS

systems. Softwares for training and test statistical models are also made available. In the end, the whole community benefits with better ASR and TTS engines.

It is well-known that collaborative networks will be critically important to business and organizations by helping to establish a culture of innovation and by delivering operational excellence. Collaborative networks have been achieved on the base of academic partnership, including several universities and some companies specialized in developing and implementing software in the university domain [31]. The collaborative networks offer the possibility to reduce the duration and the costs of development and implementation of hard to automate, vast and expensive systems [1], such as developing a ASR or TTS engine.

There are already successful multi-language audio collaborative networks like the Microsoft's YourSpeech project and the Voxforge free speech corpus and acoustic model repository. The YourSpeech platform [5] is based on crowd sourcing approaches that aims at collecting speech data and provides means that allow users to donate their speech: a quiz game and a personalized TTS system. The VoxForge project [50] was set up to collect transcribed speech to create a free speech corpus for use with open source speech recognition engines. The speech audio files are compiled into acoustic models for use with open source speech recognition engines. This work complements these previous initiatives by focusing on documenting and releasing softwares and procedures for developing engines.

The proposed VOICECONET platform is comprehensive and based on the open source concept. It aims at easing the task of developing ASR and TTS engines to any language, including the underrepresented. In summary, the framework has the following features:

- Rely on a Web-based platform that was organized for collecting resources and improving the engines accuracy. Through this platform anyone can contribute with waveform files and the system rewards them with a speaker dependent acoustic model. The users can also create a custom language model depends on the tasks where the model is going to be used.
- The adopted softwares for building and evaluating engines (HTK, Julius, MARY, etc.) are freely available. A considerable amount of time is spent to learn how to use them, and the network shares scripts to facilitate.
- The shared reports and academic papers use the concept of reproducible research [48], and the results can be replicated in different sites.

In order to collect desktop labeled data, the system architecture is based on the client/server paradigm (see Figure 4). The client accesses the platform through a website [49] and is identified by an user ID. Once there, the user is invited to record random sentences, taken from a phonetically rich set [8], in order to guarantee complete phoneme coverage. The Java Applet used for recording control was based on the VoxForge related application. This control enables the system to access the user's computer and record audio using any of the installed devices. If the user selects the submit option, the recordings automatically stop and the wave files are streamed to the server. The user can submit the recorded sentences anytime.

When enough sentences are recorded, at least 3 minutes of audio, the user may choose to generate his/her speaker dependent acoustic model. At that moment, in the server, a script

is used to process the audio and build an adapted acoustic model using the HTK tools. Then, the user may choose to download a file containing his/her speaker dependent acoustic model. Note that the quality of the model is increased by the number of recorded sentences.

Another feature allows context-dependent language model to be created. To create the language model it is necessary to input a small caps without punctuation text corpus in a simplified standard generalized markup language format. In this format, there can be only one sentence per line and each sentence is delimited by the tags $< s >$ and $< /s >$. In the sequel, a script is used to process the text and build a trigram ARPA format language model using the SRILM tools. To improve the model it is recommended that the user previously converts the input texts so that every number, date, money amount, time, ordinal number, websites and some abbreviations should be written in full.

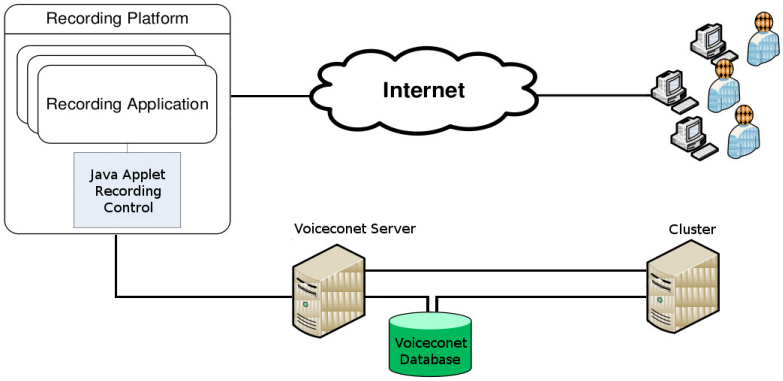

Figure 4. High level system architecture diagram.

At the time of writing of this chapter, VOICECONET is online for 1 month for the BP language. We have been receiving extremely positive feedback from the users. It was collected more than 60 minutes of audio from 15 speakers, and 5 language models were made available, adding up 10 thousand sentences. Figure 5 shows the VOICECONET platform website.

It is pedagogical to use an underrepresented language to illustrate how the global community can build and keep improving resources to all languages. Using the VOICECONET platform, the following speech-related resources for BP are currently shared: two multiple speakers audio corpora corresponding together to approximately 17 hours of audio, a phonetic dictionary with over 65 thousand words, a speaker independent HTK format acoustic model, and a trigram ARPA format language model. All these BP resources are publicly available [15] and the next section describes how they were developed.

4. Development of ASR and TTS engines: A case study

Taking again BP as an example, the most widely used corpus seems to be the Spoltech, distributed by the Linguistic Data Consortium (LDC). The LDC catalog also released the West Point Brazilian Portuguese Speech, a *read* speech database of microphone digital recordings from native and non-native speakers. These two corpora are not enough for fully developing

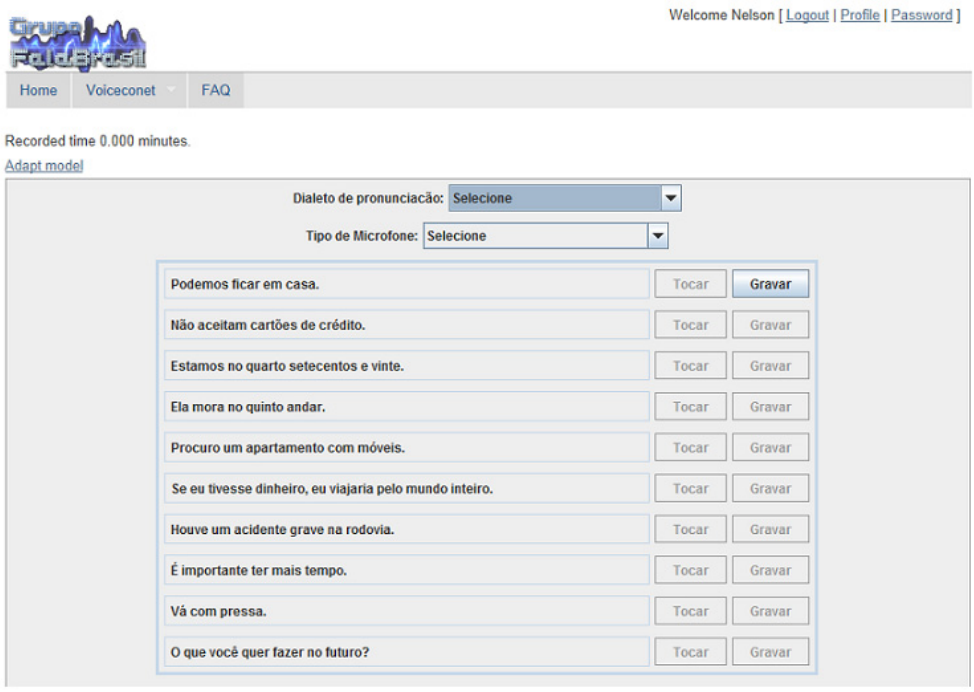

Figure 5. The VOICECONET platform website.

a state of art LVCSR systems, since they have together approximately dozen hours. For example, only the Switchboard telephone conservations corpus for English has 240 hours of recorded speech [17]. There were initiatives [35, 53] in the academy for developing corpora, but they have not established a sustainable collaboration among researchers.

Regarding end-users applications, there are important freely available systems. For instance, in Brazil, Dosvox and Motrix [47] are very popular among blind and physically-impaired people, respectively. Dosvox includes its own speech synthesizer, besides offering the possibility of using other engines. Dosvox does not provide ASR support. The current version of Motrix supports ASR only in English, which makes use of a ASR engine distributed for free by Microsoft.

There are advantages on adopting a proprietary software development model, but this work advocates the open source model. The next sections describe the resources developed in order to design speech engines for BP.

4.1. UFPAdic: A phonetic dictionary for BP

The phonetic dictionary used in this work was automatically generated using a phonetic transcription with stress determination algorithm for BP language described in Silva et al. [39]. The improvements proposed in Siravenha et al. [42] were also considered.

The proposed software (G2P converter) is based on phonological pre-established criteria and its architecture does not rely on intermediate stages, i.e., other algorithms such as syllabic division or plural identification. There is a set of rules not focus in any BP dialect for each grapheme and a specific order of application is assumed. First, the more specific rules are considered until a general case rule is reached, which ends the process. Therefore the developed G2P converter deals only with single words and does not implement co-articulation analysis between words.

The rules are specified in a set of regular expressions using the C# programming language. Regular expressions are also allowed in the definition of non-terminals symbols (e.g. #abacaxi#). The rules of the G2P converter are organized in three phases. Each phase has the following function:

- a simple procedure that inserts the non-terminal symbol # before and after each word.
- the stress phase that mark the stressed vowel of the word.
- the bulk of the system that convert the graphemes (including the stressed vowel brand) to 38 phones represented using the SAMPA phonetic alphabet [32].

Using the described G2P converter, a phonetic dictionary was created. It has 65,532 words and is called UFPAdic. These words were selected by choosing the most frequent ones in the CETENFolha corpus [7], which is a corpus based on the texts of the newspaper *Folha de S. Paulo* and compiled by NILC/*São Carlos*, Brazil. The G2P converter (executable file) and the UFPAdic are publicly available [15].

4.2. Syllabification

The developed G2P converter does not perform syllabification nor stress syllable identification. So these two tasks were implemented in a Java software publicly available at Fal [15]. The algorithm used for syllabification is described in Silva et al. [38]. The main idea of this algorithm is that all syllables have a vowel as a nucleus, and it can be surrounded by consonants or other (semi-vowels). Hence, one should locate the vowels that composes the syllable nuclei and isolate consonants and semivowels.

The original syllabification rules [38], as previously mentioned, consider the kind and the arrangement of graphemes to separate the syllables of a given word. However, there are some words whose syllabification is very difficult to perform correctly with only these two criteria, especially when such words have diphthongs, because they will require a number of very specific and well elaborated rules, in which each one will deal with just a few examples.

To overcome this difficulty, new linguistic rules, shown in Table 2, were proposed, each one not just considering the graphemes themselves, but also their stress. The first group deals with the falling diphthongs (the “vowel + glide” combination), while the second one deals with diphthongs that varies with hiatus (the “glide + vowel” combination). Due to the fact that diphthongs need this special treatment in their syllabification, it was defined that these rules must be evaluated before the 20 original ones.

The main motivation for analyzing the previously mentioned diphthongs comes from the perception of existing divergences between the scholars on such subject (like the position of a glide, inside a syllable, in the falling diphthongs, that was explained in Bisol [3]). Another point that ratifies the focus adopted in this analysis is the fact that vocalic segments, especially the ones with rising sonority, have presented many errors in the separations performed by the syllabification algorithm (in a previous analysis).

Sequence for the algorithm	Action	Example
...(a,e,o)(i(V_ton),u(V_ton))...	must be separated	sa-í-da, gra-ú-do
...(a,e,o)(i,u)...	stay in the same syllable	cãi-bra, mai-se-na
...(i,u)(a(V_ton),e(V_ton),o(V_ton))<Pont>...	must be separated	ta-man-du-á
...(i(V_ton),u(V_ton))(a,e,o)<Pont>...	must be separated	de-mo-cra-ci-a
...(i,u)(a,e,o)<Pont>...	stay in the same syllable	só-cio, cí-lio
...(i,u)(a,e,o)...	must be separated	bi-o-ma

Table 2. New syllabification rules.

In addition to that, the original rule 19 was updated to fix some errors that occurred when it was previously proposed. If the analyzed vowel is not the first grapheme in the next syllable to be formed and is followed by another vowel that precedes a consonant, then the analyzed vowel must be separated from the following graphemes. This new version of the rule 19 fixes some errors that were occurring in the syllabification of words like "teólogo", for example (the correct is "te-ó-logo", instead of "teó-lo-go", as shown in Silva et al. [38]).

Identifying the stress syllable proved to be an easier task that benefited from the fact that the developed G2P converter, in spite of not separating in syllable, was already able to identify the stressed vowel. After getting the result of the syllabification, it was then trivial to identify the syllable corresponding to the stress vowel.

4.3. LapsStory

The LapsStory corpus is based on spoken books or audiobooks. Having the audio files and their respective transcriptions (the books themselves), a considerable reduction in human resources can be achieved.

The original audio files were manually segmented to create smaller files, that were re-sampled from 44,100 Hz to 22,050 Hz with 16 bits. Currently, the LapsStory corpus consists of 8 speakers, which corresponds to 16 hours and 17 minutes of audio. Unfortunately, the LapsStory corpus cannot be completely released in order to protect the copyright of some audiobooks. Therefore, only part of the LapsStory corpus is publicly available, which corresponds to 9 hours of audio [15].

It should be noted that the acoustic environment of audiobooks is very controlled, so the audio files have no audible noise and high signal to noise ratio. Thus, when such files are used to train a system that will operate in a noisy environment, there is a problem with the acoustic mismatch. This difficulty was circumvented by the technique proposed in Silva et al.

[41], which showed that speaker adaptation techniques can be used to combat such acoustic mismatch.

4.4. LapsBenchmark

Another developed corpus is the LapsBenchmark, which aims to be a benchmark reference for testing BP systems. The LapsBenchmark's recordings were performed on computers using common (cheap) desktop microphones and the acoustic environment was not controlled.

Currently, the LapsBenchmark corpus has data from 35 speakers with 20 sentences each, which corresponds to 54 minutes of audio. It was used the phrases described in Cirigliano et al. [8]. The used sampling rate was 22,050 Hz and each sample was represented with 16 bits. The LapsBenchmark speech database is publicly available [15].

4.5. Resources required for TTS

In order to create a complete TTS system for BP, some specific resources had to be developed and procedures were executed, following the tutorials in Schröder & Trouvain [36]. The motivation for using the open-source MARY platform in this work was that it is completely written in Java and supports both concatenative and HMM-based synthesis.

Using the nomenclature adopted in the MARY framework, the task of supporting a new language can be split into the creation of a text processing module and the voice. The former enables the software to process BP text and, for example, perform the G2P conversion. The creation of a voice in this case corresponds to training HMMs using the HTS toolkit.

MARY requires a set of files for each language that it supports. This work adopted the recipe suggested by MARY for training finite state transducers (FST) [23]. The FST training procedure requires the definition of a phonetic alphabet and a list with the most frequent words in the target language. These two built files, specific for BP, are briefly described below:

- allophones.pt_BR.xml: is the phonetic alphabet, which must describe the distinctive features of each phone such as voiced/unvoiced, vowel/consonant, and others. A preliminary version of this file was developed by the authors using the SAMPA phonetic alphabet [32].
- pt_BR.dic: is a list that contains all the planned words with their corresponding phonetic transcriptions based on the phonetic alphabet previously described. These transcriptions are required to be separated into syllables and the stress syllable indicated.

After having a valid BP front end, the HTS toolkit was used to create an HMM-based back end. In order to facilitate the procedure, MARY provides the *VoiceImport* tool, illustrated in Figure 6, which is an automatic HMM training routine. For HMM training, one needs a labeled corpus with transcribed speech. It was used the speech data available with the BP demo for HTS [26], which has a total of 221 files, corresponding to approximately 20 minutes of audio. The word level transcriptions were not found at the HTS site and, for convenience to other users, were made available in electronic format at Fal [15].

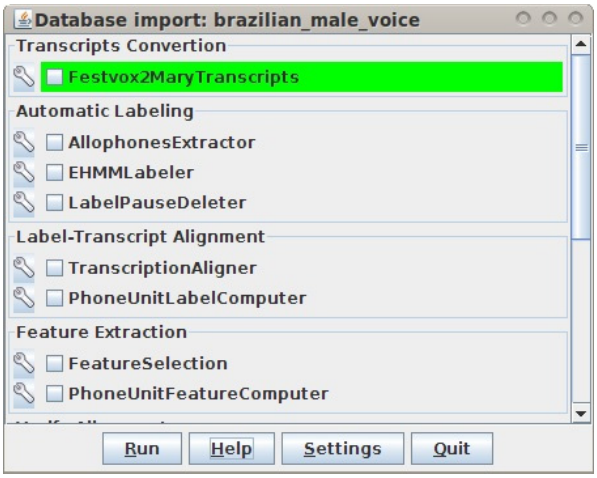

Figure 6. The GUI of the VoiceImport tool.

After this stage, the TTS system for BP is already supported by the MARY platform. All the developed resources and adopted procedures are publicly available [15].

5. Experimental results

This section presents the baseline results obtained with all the developed ASR and TTS resources for BP. The scripts and developed models were made publicly available [15]. All the experiments were executed on a computer with Core 2 Duo Intel processor (E6420 2.13 GHz) and 1 GB of RAM.

The target for the first tests presented was to understand, in practice, how the main components of a LVCSR system behave: the phonetic dictionary, acoustic model, language model and the decoder. The existing correlation between these components and the performance presented by the system are analyzed in respect of the xRT factor, as well as the WER. Finally, the quality of speech produced by the developed TTS system was compared with other synthesizers, including a commercial software.

5.1. Evaluation of the proposed syllabification rules

This experiment used 170 words in total that, aiming to delimitate the encountered problems, were divided in four contexts: falling diphthongs, rising diphthongs, diphthongs that varies with hiatus and false diphthongs. For each word, three syllabifications were analyzed and extracted from the following sources: a Portuguese web dictionary [12], the old version of the syllabification algorithm (without the new rules) and the new version of the same algorithm, with the new rules implemented.

Table 3 shows the results of the syllabification tests for each source. The divergences between the syllabifications performed by the sources and the standards for syllable separation in BP,

described in Bisol [3], were considered as errors encountered during the syllabification process of these sources.

Sources	Words	Errors	Error rates
Portuguese web dictionary	170	1	0.58%
Syllabification algorithm (old)	170	29	11.17%
Syllabification algorithm (new)	170	2	1.17%

Table 3. Error percentage analysis of the syllabification sources.

The new version of the syllabification algorithm achieved a very low amount of errors, compared to its previous version. The mentioned errors were detected in two words in the context of the falling diphthongs: "eufonia" and "ousadia". The syllable divisions "eu-fo-nia" and "ou-sa-dia" are incorrect because each one presents two vowels that should be separated to compose nuclei for two different syllables. The accepted syllabifications are: "eu-fo-ni-a" and "ou-sa-di-a".

5.2. Evaluation of the language model

This experiment evaluates the language model perplexity against the number of sentences used to train it (Figure 7). The SRILM tookit was used to build the trigram ARPA format language models. The number of sentences used to train the language models ranged between 255,830 and 1,534,980 extracted from CETENFolha and LapsStory corpora. The vocabulary was kept constant with the 65,532 distinct words present in UFPAdic.

The phrases used to measure the perplexity were separated into three test sets of ten thousand sentences each one, unseen during the training phase. The first test set was designed solely with sentences extracted from the CETENFolha corpus, the second set with sentences collected by crawling newspapers available on the Internet and the last one encompasses the phrases present in the previous sets. Note that, in terms of sentences, the CETENFolha and crawling corpus can be considered disjoint.

As expected, the perplexity tends to diminish as the number of sentences used in the training increases. This is related to the fact that the statistics of the trained models get improved as more occurrences of triples of words are registered in the database of written texts. It is important to observe, in Figure 7, the difference of perplexities measured for the data test sets. The perplexity was higher when the sentences used to evaluate the language model were collected from a different text corpus that was used to train it. It can be verified with the presented CETENFolha and crawling curves, since the crawling corpus was not used for training the models. With one thousand sentences this difference is 37%.

5.3. Evaluation of the acoustic model

The HTK software was used to build the acoustic models, according to the steps described in Young et al. [55]. Estimating a good acoustic model is considered the most challenging part of the design of an ASR system. For training an acoustic model, it is required a corpus with

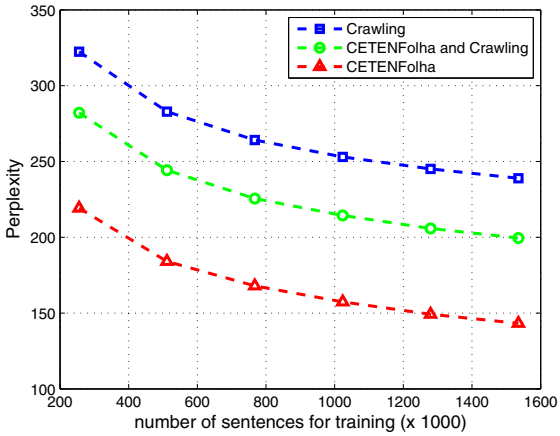

Figure 7. Perplexity against the number of sentences used to train the language model.

digitized voice, transcribed at the level of words (orthography) and/or at the level of phones. Below is a list with details about the configuration used:

- Window length: 25ms.
- Time to capture speech segments: at each 10ms (also knows as shift).
- Computed coefficients for each segment: Mel Cepstral.
- Total of coefficients: energy + 12 Mel Cepstral coefficients + first and second derivatives. On total, the computed vector for each segment has 39 coefficients.
- Acoustic modeling: continuous HMMs with 3 states left-right.
- Acoustic units (HMMs): cross-word triphone models that were built from 38 monophones and a silence model and tied with a decision tree.
- Decoder: Viterbi beam search.

The speaker independent acoustic model was initially trained using the LapsStory and West Point corpora, which corresponds to 21.65 hours of audio, and the UFPAdic. After that, the HTK software was used to adapt the acoustic model, using the maximum likelihood linear regression (MLLR) and maximum a posteriori (MAP) techniques with the Spoltech corpus, which corresponds to 4.3 hours of audio. This adaptation process was used to combat acoustic mismatches and is described in Silva et al. [41]. Both MAP and MLLR were used in the supervised training (offline) mode.

For decoding, the experiments performed in this work adopts the Julius rev.4.1.5 [**?**] and HDecode (part of HTK) [55] softwares. It was used the trigram language model trained before with 1,534,980 sentences and 14 Gaussians modeled the output distributions of the HMMs. The LapsBenchmark corpus was used to evaluate the models.

Several tests were conducted in order to evaluate the best decoding parameters for both HDecode and Julius. It was observed that the increasing on the xRT factor and recognition

accuracy can be linked to a more effective pruning of the decoder at the acoustic level. The pruning process is implemented at each time step by keeping a record of the best hypotheses overall and de-activating all hypotheses whose log probabilities fall more than a beam width below the best. Setting the beam width is thus a compromise between speed and avoiding search errors, as showed in Figure 8.

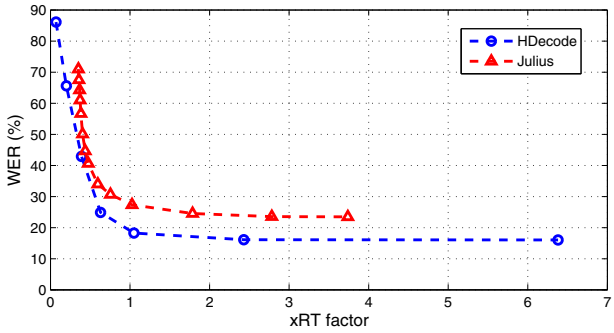

Figure 8. WER against xRT factor using the Julius and HDecode recognizers.

The tests were performed varying the beam width value from 100 to 400 and 15,000 for HDecode and Julius, respectively. It was because the WER stopped to evolve, while the xRT factor increased significantly. It was perceived that Julius can implement more aggressive pruning methods than HDecode, without significantly increasing the xRT factor. On the other hand, Julius could not achieve the same WER obtained with HDecode.

Thus the best decoding parameters for both HDecode and Julius are described in Table 4 and Table 5, respectively. For convenience, the xRT factor value was kept around one.

Parameter	Value
Pruning beam width	220
Language model scale factor	20
Word insertion penalty	22
Word end beam width	100
Number of tokens per state	8
Acoustic scale factor	1.5

Table 4. Parameters used for testing with HDecode.

5.4. Using different number of Gaussians within the acoustic model

In this test, it was considered the same acoustic and language configurations as before. However, the number of Gaussians used in the state output distributions is varied, from a single Gaussian up to 20 Gaussians, as showed in Figure 9. The LapsBenchmark corpus was used to evaluate the models.

Parameter	Value
Pruning beam width for the first pass	2,000
Pruning beam width for the second pass	200
Language model weight for the first and second passes	15
Word insertion penalties for the first and second passes	10
Score envelope width	300

Table 5. Parameters used for testing with Julius.

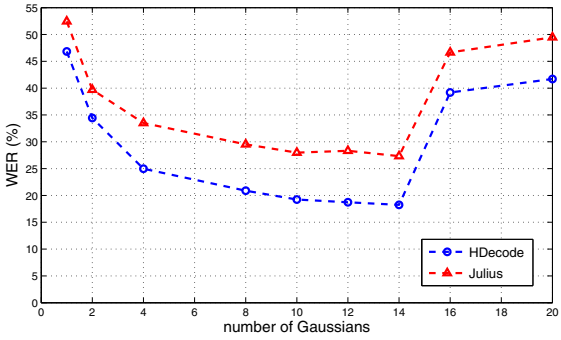

Figure 9. WER against the number of Gaussians used for training the acoustic model.

It was perceived that the computational costs added for increasing the number of Gaussians is compensated by an improvement in the decoder performance. The WER with 14-component Gaussian mixtures is 18.24% and 27.33% for HDecode and Julius, respectively. As the WER increased above 14 Gaussians, the next experiment will consider this number as a default.

5.5. Speaker dependent speech recognition

Preliminary decoding tests with the independent acoustic model adopted before were made using part of the LapsBenchmark corpus, which corresponds to 1.55 minutes of audio recorded with a selected male speaker voice. The results are showed in Figure 10.

In the sequel, a speaker dependent evaluation was performed. The same speaker acessed the VOICECONET platform and contributed with 10.8 minutes of his voice. Thus, the collected audio was used to adapt the independent acoustic model, using the MLLR and MAP adaptation techniques, according to the steps described in Young et al. [55]. The results are presented in Figure 11.

As expected, the speaker adaptation process increased the performance of all decoders. The goal of this experiment was to show that is possible to use the Internet as a collaborative network for improving the engines accuracy.

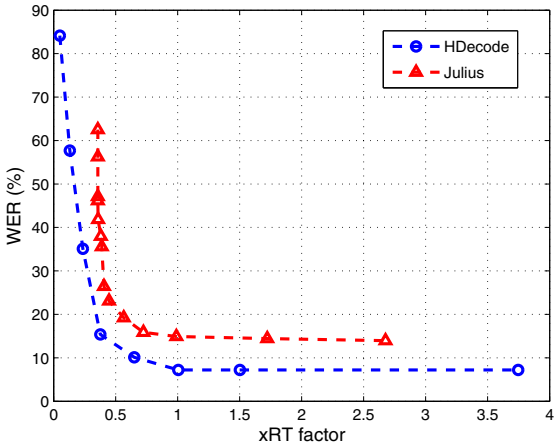

Figure 10. WER and xRT factor observed on tests of speaker independent recognition.

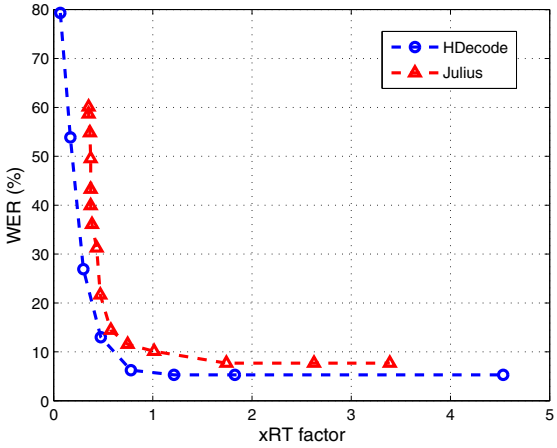

Figure 11. WER and xRT factor observed on tests of speaker dependent recognition.

5.6. Evaluation of the TTS system

This experiment evaluates the TTS system developed in this work. Experiments were also made with two other BP TTS systems: Liane and Raquel. Liane is a MBROLA synthesizer [13] and Raquel is a state-of-art commercial software of Nuance [29]. Both are concatenative diphone synthesizers and were used only for comparison purposes.

The evaluation process was carried out in two stages: segmental intelligibility and pleasantness. The pleasantness of speech was analyzed in respect of the MOS protocol and the WER was used to mensure the intelligibility. Twenty sentences (about 5 seconds each one) collected from the Internet were used to perform the tests. Thus, the sixty audio files were

randomly played and the listener had to give a note for pleasantness and repeat what he/she heard. On total, ten people were interviewed, and none of which has formal education in speech area. The results are showed in Figure 12 and 13.

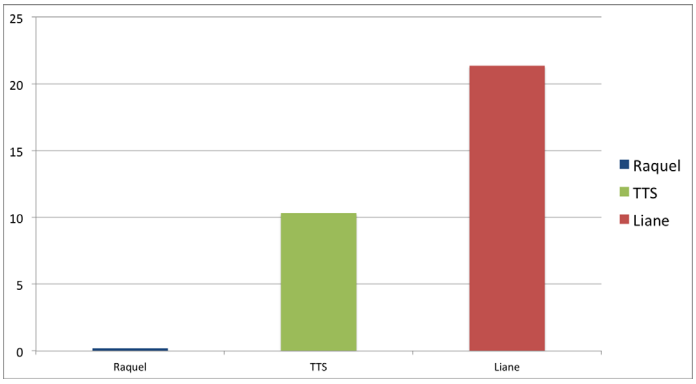

Figure 12. WER observed for each TTS software.

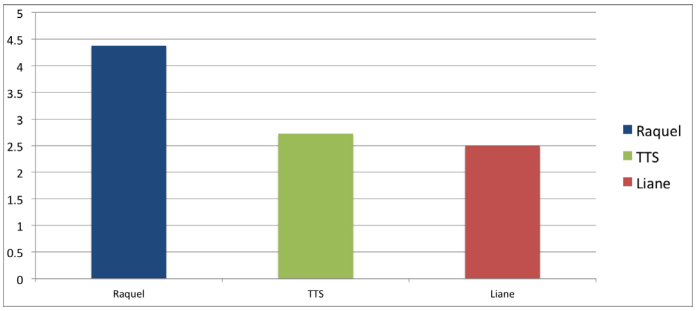

Figure 13. The average pleasantness for each TTS software.

In pleasantness evaluation, the developed TTS system and Liane were considered slightly annoying, while Raquel was evaluated as a good synthesizer. It was observed that even the most developed systems still are less natural than the human voice. In objective test, the TTS system (with WER around 10%) outperforms Liane. Raquel achieved again the best result.

6. Conclusions

This chapter advocates the potential of using speech-enabled applications as a tool for increasing social inclusion and education. This is specially true for users with special needs. In order to have such applications for underrepresented languages, there is still a lot of work to be done. As discussed, one of the main problems is the lack of data for training and testing the systems, which are typically data-driven. So, in order to minimize cost, this work presents the VOICECONET, a collaborative framework for building applications using speech recognition and synthesis technologies. Free tools and resources are made publicly available [15], which include a complete TTS and ASR systems for BP.

Future work includes expanding both the audio and text databases, aiming at reaching the performance obtained by ASR for English and Japanese, for example. The phonetic dictionary refinement is another important issue to be addressed, considering the existing dialectal variation in Brazil. In parallel, improving the free TTS for BP and develop prototypes for groups of people that are not usually the main target of the assisting technology industry and that require special attention. And finally, one important goal is to help groups that use the network for developing resources for languages other than BP.

Acknowledgements

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, project no. 560020/2010-4.

Author details

Nelson Neto, Pedro Batista and Aldebaro Klautau
Federal University of Pará (UFPA), Signal Processing Laboratory (LaPS) – http://www.laps.ufpa.br, Belém – PA – Brazil

7. References

[1] Agranoff, R. [2006]. Inside collaborative networks: Ten lessons for public managers, *Public Administration Review, 66* pp. 56–65.

[2] Antoniol, G., Fiutem, R., Flor, R. & Lazzari, G. [1993]. Radiological reporting based on voice recognition, *Human-computer interaction. Lecture Notes in Computer Science, 753* pp. 242–253.

[3] Bisol, L. [2005]. *Introdução a Estudos de Fonologia do Português Brasileiro*, Porto Alegre: EDIPUCRS.

[4] Black, A., Taylor, P. & Caley, R. [1999]. *The Festival Speech Synthesis System*, The University of Edinburgh, System Documentation, Edition 1.4.

[5] Calado, A., Freitas, J., Silva, P., Reis, B., Braga, D. & Dias, M. S. [2010]. Yourspeech: Desktop speech data collection based on crowd sourcing in the internet, *International Conference on Computational Processing of Portuguese - Demos Session* .
URL: *http://pt.yourspeech.net*

[6] Caseiro, D., Trancoso, I., Oliveira, L. & Viana, C. [2002]. Grapheme-to-phone using finite-state transducers, *IEEE Workshop on Speech Synthesis* pp. 215–218.

[7] CET [2012]. CETENFolha Text Corpus. Visited in March.
URL: *www.linguateca.pt/CETENFolha/*

[8] Cirigliano, R., Monteiro, C., Barbosa, F., Resende, F., Couto, L. & Moraes, J. [2005]. Um conjunto de 1000 frases foneticamente balanceadas para o Português Brasileiro obtido utilizando a abordagem de algoritmos genéticos, *XXII Simpósio Brasileiro de Telecomunicações* pp. 544–549.

[9] Damper, R. [2001]. *Data-Driven Methods in Speech Synthesis*, Kluwer Academic.

[10] Davis, S. & Merlmestein, P. [1980]. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, *IEEE Transactions on Acoustics, Speech, and Signal Processing, 28* (4): 357–366.

[11] Deshmukh, N., Ganapathiraju, A. & Picone, J. [1999]. Hierarchical search for large-vocabulary conversational speech recognition, *IEEE Signal Processing Magazine* pp. 84–107.
[12] Dic [2012]. Portuguese Web Dictionary. Visited in March.
URL: *www.dicionarioweb.com.br*
[13] Dutoit, T., Pagel, V., Pierret, N., Bataille, F. & Vrecken, O. [1996]. The MBROLA project: Towards a set of high quality speech synthesizers free of use for non commercial purposes, *Proceedings of the 4th International Conference of Spoken Language Processing* pp. 1393–1396.
[14] Eskenazi, M. [2009]. An overview of spoken language technology for education, *Speech Communication, 51* (10): 832–844.
[15] Fal [2012]. Research Group FalaBrasil. Visited in March.
URL: *www.laps.ufpa.br/falabrasil/index_en.php*
[16] Fidalgo-Neto, A., Tornaghi, A., Meirelles, R., Berçot, F., Xavier, L., Castro, M. & Alves, L. [2009]. The use of computers in Brazilian primary and secondary schools, *Computers & Education, 53* (3): 677–685.
[17] Godfrey, J., Holliman, E. & McDaniel, J. [1992]. SWITCHBOARD: Telephone speech corpus for research and development, *IEEE International Conference on Acoustics, Speech and Signal Processing, 1* pp. 517–520.
[18] Huang, X., Acero, A. & Hon, H. [2001]. *Spoken Language Processing*, Prentice Hall.
[19] Huggins-Daines, D., Kumar, M., Chan, A., Black, A. W., Ravishankar, M. & Rudnicky, A. [2006]. Pocketsphinx: A free, real-time continuous speech recogntion system for hand-held devices, *Proceedings of ICASSP* pp. 185–188.
[20] IBG [2010]. Census Demographic Profiles.
URL: *www.ibge.gov.br/home/estatistica/populacao/censo2010/*
[21] JSA [2012]. Java Speech API. Visited in March.
URL: *java.sun.com/products/java-media/speech/*
[22] Juang, H. & Rabiner, R. [1991]. Hidden Markov models for speech recognition, *Technometrics, 33* (3): 251–272.
[23] Kornai, A. [1999]. Extended Finite State Models of Language, Cambridge University Press.
[24] Lee, A., Kawahara, T. & Shikano, K. [2001]. Julius - an open source real-time large vocabulary recognition engine, *Proc. European Conf. on Speech Communication and Technology* pp. 1691–1694.
[25] Lee, C. & Gauvain, J. [1993]. Speaker adaptation based on MAP estimation of HMM parameters, *IEEE International Conference on Acoustics, Speech and Signal Processing, 2* pp. 558–561.
[26] Maia, R., Zen, H., Tokuda, K., Kitamura, T. & Resende, F. [2006]. An HMM-based Brazilian Portuguese speech synthetiser and its characteristics, *Journal of Communication and Information Systems, 21* pp. 58–71.
[27] Mea [1996]. P.800 - ITU: Methods for Subjective Determination of Transmission Quality.
URL: *www.itu.int/rec/T-REC-P.800-199608-I/en*
[28] MLD [2012]. Microsoft Development Center. Visited in March.
URL: *www.microsoft.com/portugal/mldc/default.mspx*

[29] Nua [2012]. Nuance Communications, Inc. Visited in March.
URL: *www.nuance.com*

[30] O'Harea, E. & McTearb, M. [1999]. Speech recognition in the secondary school classroom: an exploratory study, *Computers & Education, 33* (1): 27–45.

[31] Sabau, G., Bologa, R., Bologa, R. & Muntean, M. [2009]. Collaborative network for the development of an informational system in the SOA context for the university management, *International Conference on Computer Technology and Development, 1* pp. 307–311.

[32] SAM [2012]. SAMPA Phonetic Alphabet. Visited in March.
URL: *www.phon.ucl.ac.uk/home/sampa/*

[33] SAP [2012]. Microsoft Speech API. Visited in March.
URL: *www.microsoft.com/speech/*

[34] Saz, O., Yin, S.-C., Lleida, E., Rose, R., Vaquero, C. & Rodríguez, W. [2009]. Tools and technologies for computer-aided speech and language therapy, *Speech Communication, 51* (10): 948–967.

[35] Schramm, M., Freitas, L., Zanuz, A. & Barone, D. [2000]. A Brazilian Portuguese language corpus development, *International Conference on Spoken Language Processing, 2* pp. 579–582.

[36] Schröder, M. & Trouvain, J. [2001]. The German text-to-speech synthesis system MARY: A tool for research, development and teaching, *International Journal of Speech Technology, 6* (4): 365–377.

[37] Sealea, J. & Cooperb, M. [2010]. E-learning and accessibility: An exploration of the potential role of generic pedagogical tools, *Computers & Education, 54* (4): 1107–1116.

[38] Silva, D., Braga, D. & Resende, F. [2008]. Separação das sílabas e determinação da tonicidade no Português Brasileiro, XXVI Simpósio Brasileiro de Telecomunicações pp. 1–5.

[39] Silva, D., de Lima, A., Maia, R., Braga, D., de Moraes, J., de Moraes, J. & Resende, F. [2006]. A rule-based grapheme-phone converter and stress determination for Brazilian Portuguese natural language processing, *VI International Telecommunications Symposium* pp. 992–996.

[40] Silva, P., Batista, P., Neto, N. & Klautau, A. [2010]. An open-source speech recognizer for Brazilian Portuguese with a windows programming interface, *Computational Processing of the Portuguese Language, Springer, 6001* pp. 128–131.

[41] Silva, P., Neto, N. & Klautau, A. [2009]. Novos recursos e utilização de adaptação de locutor no desenvolvimento de um sistema de reconhecimento de voz para o Português Brasileiro, *XXVII Simpósio Brasileiro de Telecomunicações* pp. 1–6.

[42] Siravenha, A., Neto, N., Macedo, V. & Klautau, A. [2008]. Uso de regras fonológicas com determinação de vogal tônica para conversão grafema-fone em Português Brasileiro, *7th International Information and Telecommunication Technologies Symposium* pp. 1–6.

[43] Siravenha, A., Neto, N., Macedo, V. & Klautau, A. [2009]. A computer-assisted learning software using speech synthesis and recognition in Brazilian Portuguese, Interactive Computer Aided Blended Learning pp. 1–5.

[44] Stolcke, A. [2002]. SRILM - an extensible language modeling toolkit, *International Conference on Spoken Language Processing* pp. 901–904.

[45] Taylor, P. [2009]. *Text-To-Speech Synthesis*, Cambridge University Press.

[46] Teixeira, A., Oliveira, C. & Moutinho, L. [2006]. On the use of machine learning and syllable information in European Portuguese grapheme-phone conversion, *Computational Processing of the Portuguese Language, Springer, 3960* pp. 212–215.

[47] UFR [2012]. Accessibility Projects of NCE/UFRJ. Visited in March. URL: *http://intervox.nce.ufrj.br/*

[48] Vandewalle, P., Kovacevic, J. & Vetterli, M. [2009]. Reproducible research in signal processing - what, why, and how, *IEEE Signal Processing Magazine, 26* pp. 37–47.

[49] Voi [2012]. VOICECONET. Visited in March. URL: *www.laps.ufpa.br/falabrasil/voiceconet/*

[50] Vox [2012]. VoxForge.org. Visited in March. URL: *www.voxforge.org*

[51] Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P. & Woelfel, J. [2004]. *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*, Sun Microsystems, TR-2004-139.

[52] Wang, T.-H. [2010]. Web-based dynamic assessment: Taking assessment as teaching and learning strategy for improving students' e-learning effectiveness, *Computers & Education, 54* (4): 1157–1166.

[53] Ynoguti, C. A. & Violaro, F. [2008]. A Brazilian Portuguese speech database, *XXVI Simpósio Brasileiro de Telecomunicações* pp. 1–6.

[54] Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T. & Kitamura, T. [1999]. Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis, *Proc. of EUROSPEECH, 5* pp. 2347–2350.

[55] Young, S., Ollason, D., Valtchev, V. & Woodland, P. [2006]. *The HTK Book*, Cambridge University Engineering Department, Version 3.4.