J Braz Comput Soc (2011) 17: 53–68
DOI 10.1007/s13173-010-0023-1

ORIGINAL PAPER

Free tools and resources for Brazilian Portuguese speech recognition

Nelson Neto · Carlos Patrick · Aldebaro Klautau · Isabel Trancoso

Received: 5 July 2010 / Accepted: 19 October 2010 / Published online: 4 November 2010

Abstract An automatic speech recognition system has modules that depend on the language and, while there are many public resources for some languages (e.g., English and Japanese), the resources for Brazilian Portuguese (BP) are still limited. This work describes the development of resources and free tools for BP speech recognition, consisting of text and audio corpora, phonetic dictionary, grapheme-to-phone converter, language and acoustic models. All of them are publicly available and, together with a proposed application programming interface, have been used for the development of several new applications, including a speech module for the OpenOffice suite. Performance tests are presented, comparing the developed BP system with a commercial software. The paper also describes an application that uses synthesis and speech recognition together with a natural language processing module dedicated to statistical machine translation. This application allows the translation of spoken conversations from BP to English and vice versa. The resources make easier the adoption of BP speech technologies by other academic groups and industry.

Keywords Speech recognition · Brazilian Portuguese · Grapheme-to-phone conversion · Application programming interface · Speech-based applications

N. Neto (✉) · C. Patrick · A. Klautau
Federal University of Pará, Augusto Correa, 1, Belém, Brazil
e-mail: nelsonneto@ufpa.br

C. Patrick
e-mail: patrickalves@ufpa.br

A. Klautau
e-mail: aldebaro@ufpa.br

I. Trancoso
IST/INESC-ID, Alves Redol, 9, Lisbon, Portugal
e-mail: isabel.trancoso@inesc-id.pt

1 Introduction

Speech processing includes several technologies, among which automatic speech recognition (ASR) [1, 2] and text-to-speech (TTS) [3, 4] are the most prominent. TTS systems are software modules that convert natural language text into synthesized speech [5]. ASR can be seen as the TTS inverse process in which the digitized speech signal is converted into text. In spite of problems such as limited robustness to noise, ASR also has its market, which, according to Opus Research, topped one billion dollars for the first time in 2006 and is expected to reach US$ 3 billions in 2010 with niches such as medical reporting and electronic health care record. Dominated in the past by companies specialized in ASR, the market currently has players such as Microsoft and Google, heavily investing in supporting ASR (and TTS) on Windows [6] and Chrome [7], for example. This work presents the results of an ambitious project, which aims at helping the academy and software industry in the development of speech science and technology focused in BP.

ASR is a data-driven technology that requires a relatively large amount of labeled data. The researchers rely on public corpora and other speech-related resources to expand the state of the art. Some research groups have proprietary speech and text corpora [8–10]. For European Portuguese (EP), the main resource collection efforts have targeted Broadcast News (BN), aiming at automatic captioning applications for the deaf community. The manually labeled BN corpus contains around 60 hours of audio, but even with this limited size, it has already allowed the deployment of a fully automatic subtitling system [11], on line at the public TV channel since March 2008. Other speech corpora have been collected for other domains: BDPublico [12] (EP database equivalent to the Wall Street Journal corpus [13]), CORAL [14] (map-task dialog corpus),

and LECTRA [15] (university classroom lectures, currently comprising 27 hours of audio).

For BP, 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 nonnative speakers. These two corpora are not enough for fully developing a state of art large vocabulary continuous speech recognition (LVCSR) systems in BP. For example, training an ASR with the maximum mutual information (MMI) criterion [16] requires many hours of audio data for training, otherwise the MMI estimation will not be effective when compared to the conventional maximum likelihood criterion. Besides the scarcity of data, there are no publicly available scripts (or software *recipes*) to design BP baseline systems. These recipes considerably contribute towards shortening the development process.

Hence, two enabling factors for developments in ASR are data and scripts. In response to this need, the *FalaBrasil* project [17] was initiated in 2009. It aims at developing and deploying resources and software for BP speech processing. The public resources allow one to establish baseline systems and reproduce results across different sites. Due to aspects such as the increasing importance of reproducible research [18], the FalaBrasil project achieved good visibility and is now fomented by a very active open-source community. Most of the currently available resources are for ASR and allow composing a complete LVCSR, which is the subject of this work. A TTS system is also under development [19] and is used in the translation example in Sect. 8, but is not detailed here.

This work follows two guidelines for promoting a faster dissemination of speech technologies in BP:

- in the academy, to increase the synergy among research groups working in BP: availability of public domain resources for ASR and TTS. Both technologies are data-driven and depend on relatively large labeled corpora, which are needed for the development of state-of-art systems;
- in the software industry, to help programmers and entrepreneurs to develop speech-enabled systems: availability of *engines* (for ASR and TTS), preferably free and with licenses that promote commercialization, and tutorials and how-to's that target professionals without specific background in speech processing. In the latter case, the existence of application programming interfaces (APIs) is crucial because very few programmers have formal education in areas such as digital signal processing and hidden Markov models.

With respect to the API, the most widely used in the industry is SAPI, the speech API from Microsoft [20]. There are other alternatives such as JSAPI (Java Speech API) from Sun Inc. These APIs specify a cross-platform interface to support command and control recognizers, dictation systems, and speech synthesizers [21]. 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. Microsoft also provides ASR engines and software development tools for BP and EP [22]. However, these systems are not open-source code.

Most previous work in ASR for BP was restricted to systems using a small vocabulary (e.g., [23, 24]). The development of a speaker-independent LVCSR for BP with a vocabulary of more than 60 thousand words is discussed in [25], where the authors target the creation of a mapped phonetic dictionary and the improvement of the language model. The results were obtained with a relatively small amount of audio data extracted from the Spoltech corpus.

In [9], a proprietary audio corpus recorded by a single speaker (the amount of audio was not reported) was used to train the stochastic speaker-dependent acoustic models, and a textual database was developed to train language models based on n-gram. The best accuracy rate obtained for the 60 thousand words system was 81% when recognizing sentences with 9 to 12 words, with perplexities ranging between 250 to 350 and processing times less than one minute per sentence. All the tests were executed on a computer with a Dual Intel processor (XeonTM 3.0 MHz) and 2 GB of RAM.

Dictation is a good task to stress test LVCSR systems [2]. There are many commercial softwares that have good performance in dictation for several languages. For BP, the only commercial desktop software is the IBM ViaVoice, which was discontinued. In spite of being relatively outdated, ViaVoice was used for comparison in this work. In the academy, recently a broadcast news LVCSR system originally developed for EP was ported to BP and achieved a word error rate of approximately 25% [26].

A motivation of this work is to complement these previous initiatives and release resources of a state of art LVCSR for BP [17], with the exception of the materials protected by copyright. The implemented system establishes a baseline, enables the comparison of results among research groups [18], and promotes the development of speech-enabled software via the proposed API. In summary, the contributions of this work are:

- Resources for the training and test stages of ASR systems: a text corpus based on ten daily Brazilian newspapers, automatically formatted and collected from the Internet; two multiple speakers audio corpora corresponding together to approximately 16.5 hours of audio.
- A grapheme-to-phone converter with stress determination for BP. The resulting phonetic dictionary has over 65 thousand words.

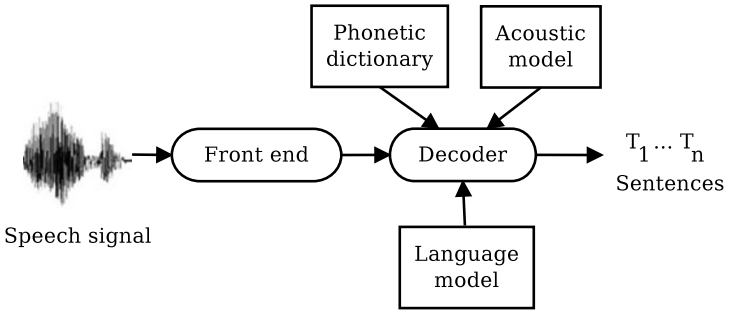


Fig. 1 The main constituent blocks of a typical ASR system

– An API that hides from the user the low-level details of the decoder operation. The proposed API contains a Microsoft SAPI XML grammar converter for easing the support of ASR.
– As a proof of concept, a speech-enabled machine translation system from BP to English and vice-versa. The goal is to allow a spoken dialog between native speakers of these languages via automatic translation.

The remainder of the paper is organized as follows. Section 2 presents a description of ASR system. Section 3 describes the linguistic resources for BP developed and used in this work such as corpora and phonetic dictionary. Section 4 describes the adopted front end and HMM-based acoustic modeling. Section 5 shows how the language model was built. Section 6 describes the API to operate the recognizer. The baseline results are presented in Sect. 7. Section 8 describes applications of the developed system and resources. Finally, Sect. 9 summarizes our conclusions and addresses future works.

2 Statistical speech recognition

The typical ASR system adopts a statistical approach based on *hidden Markov models* (HMMs) [27, 28] and is composed by four main blocks: front end, acoustic model, language model, and decoder, as indicated in Fig. 1, which also shows the phonetic dictionary.

2.1 The main blocks of an 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 an $L \times T$ matrix $\mathbf{X}$, which represents a complete sentence.

There are several alternatives to parameterize the speech waveforms. Although, the Mel-frequency cepstral coefficients (MFCCs) analysis have been proven to be effective and used pervasively as the direct input to the ASR back end [2].

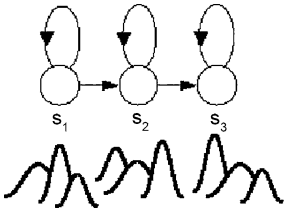


Fig. 2 Pictorial representation of a left-right continuous HMM with 3 states and a mixture of Gaussians per state

The language model provides the probability $p(\mathcal{T})$ of observing a sentence $\mathcal{T} = [w_1, \ldots, w_P]$ of P words. Conceptually, the goal is to find the sentence $\mathcal{T}^*$ that maximizes the posterior

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

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

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

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

Because of the large number of possible sentences, (1) cannot be calculated independently for each candidate sentence. Therefore, ASR systems use data structures such as lexical trees and are hierarchical, breaking sentences into words, and words into *basic units* as phones [2]. The search for $\mathcal{T}^*$ is called decoding, and, in most cases, hypotheses are pruned (i.e., some sentences are discarded, and (1) is not calculated for them) to make the search feasible [29, 30].

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 Fig. 2. The HMM topology is "left-right," in which the only valid transitions are loops or to the next state.

Table 1 Example of phone transcription using context-dependent models. The phones are represented using the SAMPA phonetic alphabet [32]

Model	Transcription
Monophones	sil u~ sp d E s sil
Word-internal	sil u~ sp d+E d-E+s E-s sil
Cross-word	sil sil-u~+d sp u~-d+E d-E+s E-s+sil sil

Two major problems in acoustical modeling are the phone variability due to coarticulation and insufficient data to estimate the models. Sharing (or tying) aims to combat the latter problem by improving the robustness of the models. In many systems, sharing is implemented at the state level, i.e., the same state can be shared by different HMMs.

Ideally, the phones would have unique articulatory and acoustic correlates. However, the acoustic properties of a given phone can change as a function of the phonetic environment. This contextual influence, known as coarticulation, is responsible for the overlap of phonetic information in the acoustic signal from segment to segment and for the smearing of segmental boundaries [31]. Hence, coarticulation motivates the adoption of context-dependent models in ASR such as the word-internal and cross-word triphones [2].

The cross-word triphone models take into account the coarticulation effects between the words boundaries, and the word-internal models ignore the words boundaries. For example, in Table 1, the sentence "um dez" is converted into context-dependent models.

Data scarcity also affects the language model that estimates

$$P(\mathcal{T}) = P(w_1, w_2, \ldots, w_P) = P(w_1)P(w_2|w_1)\ldots P(w_P|w_1, w_2, \ldots, w_{P-1}).$$

It is impracticable to robustly estimate the conditional probability $P(w_i|w_1, \ldots, w_{i-1})$, even for moderate values of i. So, the language model for LVCSR consists of an n-gram model, which assumes that the probability $P(w_i|w_1, \ldots, w_{i-1})$ depends only on the $n-1$ previous words. For example, the probability $P(w_i|w_{i-2}, w_{i-1})$ expresses a trigram language model.

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 [33]. The adaptation of the acoustic model is also important [34].

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.

The maximum likelihood linear regression (MLLR) is the adaptation by linear transformations. This technique computes a set of transformations that will reduce the mismatch between an initial model set (the speaker-independent model) and the adaptation data provided by the user [35]. The effect of these transformations is to shift the component means in the initial system so that each state in the HMM system is more likely to generate the adaptation data. Model adaptation can also be accomplished using a maximum a posteriori (MAP) or Bayesian approach [35].

2.2 Evaluation metrics

In most ASR applications (including dictation) the figure of merit of an ASR system is the word error rate[1] (WER). Given that in general the correct and recognized transcriptions have a different number of words, they are aligned through *dynamic programming* [36]. An *edit distance* is then used to account for deletions, insertions, and substitutions, which are all taken in account when computing WER [2].

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 $\mathbf{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_S\}$ composed of S sentences. Independence among the sentences is assumed, which leads to $p(\mathbf{T}) = p(\mathcal{T}_1), \ldots, p(\mathcal{T}_S)$. Two measures are derived from this probability, perplexity and cross-entropy [2]. The cross-entropy $H_p(\mathbf{T})$ is defined as

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

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

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

Lower cross-entropies and perplexities indicate less uncertainty in predicting the next word and, for a given task

[1] Within dialogue applications like dialog management, it is possible to evaluate the system according to parameters as task completion.

(vocabulary size, etc.), typically indicate a better language model.

2.3 Used tools

The HTK software [37] was used to build and adapt the acoustic models presented in this work. The SRI Language Modeling Toolkit (SRILM) was used to build the *n*-gram ARPA format language models. The SRILM [38] is a toolkit for building and applying statistical language models. This software also enables the use of many *n*-gram smoothing algorithms.

For decoding, the system developed in this work adopts Julius rev.4.1.5 [39], which is typically capable of operating in real time and is open-source. Experiments were also made with another decoder: HDecode (part of HTK). The current version of HDecode [37] can be run in full decoding where an *n*-gram language model (up to trigram) is used for recognition but has no support for real-time operation. In this paper HDecode was used only for comparison purposes.

After this description of ASR, the next section describes the resources developed for BP and the third party corpora used to perform the experiments.

3 Linguistic resources for BP

In order to increase the number of ASR resources for BP, some specific resources were built. First, this section presents the developed phonetic dictionary, text and speech corpora. Finally, the Spoltech and West Point corpora are described. The Spoltech and West Point speech corpora are protected by copyright and can be purchased from LDC. The major revision and corrections that were performed on these corpora along this research is documented at [17].

3.1 UFPAdic: a phonetic dictionary for BP

An essential building block for services involving speech processing techniques is the correspondence between the orthography and the pronunciation(s). For instance, in order to develop LVCSR for BP, one needs a pronunciation (or phonetic) dictionary, which maps each word in the lexicon to one or more phonetic transcriptions (pronunciations). 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 preexistent G2P module. However, the task of designing a G2P module is not trivial, and several techniques have been adopted over the last decade [40, 41].

This work presents a G2P converter with stress determination for BP that is based on a set of rules described in [42]. The rules did not focus in any BP dialect. One advantage of rule-based G2P converters is that the lexical alignment is not necessary [42], which is a requirement of some approaches based on machine learning.

The proposed conversion is based on phonological pre-established criteria, 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 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.

The general format of each dictionary entry suggested by the HTK software [37] is illustrated by the example below:

```
leite l e j tS i sp
```

Therefore the developed G2P converter deals only with single words and does not implement coarticulation 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 nonterminal 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 nonterminal symbol # before and after each word.
- the stress phase consists of 29 rules that mark the stressed vowel of the word.
- the bulk of the system, which consists of 140 rules that convert the graphemes (including the stressed vowel brand) to 38 phones represented using the SAMPA phonetic alphabet [32].

The following example illustrates the regular expression specification used to analyze the word “abacaxi” which identifies *i* as the stressed vowel. First, the “Regex” object is created with the pattern to be found within the word analyzed. After that, the “Match” object receives the response pattern comparison with the word presented. Being true, the stressed vowel is determined; otherwise, other rules are tested until they run out the possibilities and the general case is applied. An example is shown below:

```
Regex rule_8=new Regex("[^aeiou][iu][#]");
Match m8 = rule_8.Match(word);
if(m8.Success) {
   index_strVw = m8.Index+1;
   strVw = word.Substring(index_strVw,1);
   break;
}
```

The code presented above describes the rule 8 applied for the determination of the stressed vowel [42] and can be explained as follows. The end of the word is indicated by the symbol #. So, the grapheme *i* or *u* is the last character of the word, and the next to the last character cannot be a vowel. The “Match” object index is a pointer for the first component

of the pattern analysis, in this case, the character that is not a vowel. Finally, the last character is defined as the stressed vowel. For example, considering the last syllable ⟨*xi*⟩ in the word "abacaxi", since this case falls into rule 8, the stressed vowel is the letter *i*.

Based on this information, the next phase is the G2P conversion, which follows the sequential order of the word (left-right). The following example presents one of the rules applied for the transcription, where the letter *x* is converted into the corresponding phone *S*.

```
letter = word.Substring(index,1);
Regex idX = new Regex("x");
Match gX = idX.Match(letter);
if(gX.Success)
{
   phone[index] = "S";
   index++;
}
```

Note that default rules need to be the last, and in some cases in which the contexts of different rules overlap partially, the most specific rule needs to be applied first. Besides, the presence of a stressed vowel changes the G2P converter interpretation, e.g.:

```
<e(V_ton)><l><C-h,Pont>  - [E]

<e(V_aton)><l><C-h,Pont> - [e]
```

distinguishes an open vowel *e* represented as phone *E* from *e*.

The conversion is temporarily stored in an array of strings until the last G2P converter step, which removes the graphemes in order to produce a sequence of phones. Finally, the word and its corresponding G2P conversion are written in the form:

```
abacaxi a b a k a S i sp
```

which is the format suggested by the HTK software [37], where the phone *sp* (short pause) must be added to the end of every pronunciation.

During the research, some rules proposed by [42] were improved, and others were added. A summary of the added or modified rules can be seen in Table 2. For example, originally, rules used to treat the nasal vowels *a* and *u* did not take into account whether the following grapheme is a vowel or consonant. However, it was found that this distinction is important for the transcription process. For instance, according to [42], the word "adotando", where the stressed nasal vowel *a* is followed by the consonant *d*, would be converted to

```
adotando a d o t a~ n d u sp
```

where the transcription of the letter *n* was not skipped. Using the improved rule, described in the first row of Table 2, the word was converted to

Table 2 New rules for graphemes [i, a, u, x]

Letter	Rule	Sequence for the algorithm	Phone
a	3	... (a(V_ton))(m,n)(V,h)...	[a∼]
i	1	Exception: gratuito(a)	[j]
u	2	...(u(m,n))(C-h)...	[u∼]
u	4	...(u)(m,n)(V,h)...	[u∼]
u	5	...(V-u)(u)...	[w]
u	6	...(q,g)(u)(a)...	[w]
u	7	...(g)(u)(o)...	[w]
x	1	...(V,C-f,m)(i)(x)...	[S]
x	2	...(f,m)(i)(x)...	[k s]
x	3	...((W_bgn)e,ê)(x)(V,C_v)...	[z]
x	4	...((W_bgn)ine)(x)(o,C_v)...	[k s]
x	5	...((W_bgn)ine)(x)(a,e,i)...	[z]
x	6	...((W_bgn)(e,ê,ine))(x)(C_uv)...	[s]
x	7	...((W_bgn)e)(x)(Hf)(V,C_v)...	[z]
x	8	...((W_bgn)e)(x)(Hf)(C_uv)...	[s]
x	9	...(V-e)(x)(Hf)(Ltr)...	[k z]
x	10	...(V-e)(x)(V)...	[k s]
x	11	...(b,f,m,p,v,x)(e)(x)(V)...	[S]
x	12	...(V)(e)(x)(V)...	[z]
x	13	...(C-b,f,m,p,v,x)(e)(x)(V)...	[k s]
x	14	...((W_bgn)x)...	[S]
x	15	...(e,é,ê)(x)(C)...	[s]
x	16	...(x)(Pont)...	[k s]
x	17	...(x)...	[S]

```
adotando a d o t a~ d u sp
```

It was also observed the absence of rules to model the grapheme *u* as a semi-vowel, represented here by the phone *w*. Accordingly, three new rules were drawn up and incorporated before the general case rule applied to grapheme *u*. For instance, according to [42], the word "quatro", would be converted to

```
quatro k u a t r u sp
```

where the diphthong formed by the grapheme sequence ⟨*qu*⟩ is modeled as the diphone sequence ⟨*ku*⟩. Now, using the rule 6 shown in Table 2, the word was converted to

```
quatro k w a t r u sp
```

Another point that proved to be important was the processing of grapheme *x*. According to [43], the letter *x* represents the most variable consonant sound in Portuguese. Since [42] uses an exception word list to convert the grapheme *x*, this work contributes with 17 new rules. The theoretical support to improve and add those rules was extracted from [43].

Regarding the rules for determining the stressed vowel, the only modification with respect to [42] was the treatment of the monosyllable "que".

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 [44], 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 [17].

3.2 LapsNews

The language models of recognition systems are typically built using interpolated models of speech corpora word level transcriptions and newspaper texts. Our initial newspaper corpus was CETENFolha, which was expanded by fully automatizing the collection of crawling ten daily Brazilian newspapers available on the Internet. After obtaining the text files, some examples of the formatting operations are:

- Removal of punctuation marks and tags ([ext], [t], [a], and others).
- Conversion to lowercase letters.
- Expansion of numbers and acronyms.
- Correction of grammatically incorrect words.

An example of the result of these operations is given below:

```
Before: A <<caixa>> do Senado tem R$ 2.000
After: a caixa do senado tem dois mil reais
```

The resulting BP text corpus has nearly 672 thousand formatted sentences and is called LapsNews. The LapsNews corpus is publicly available [17].

3.3 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 resampled from 44,100 Hz to 22,050 Hz with 16 bits. Currently, the LapsStory corpus consists of 7 speakers, which corresponds to 15 hours and 42 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 [17].

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

3.4 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. We used the phrases described in [46]. The used sampling rate was 22,050 Hz, and each sample was represented with 16 bits. The LapsBenchmark speech database is publicly available [17].

3.5 Spoltech system

The Spoltech corpus [47] was created by the Federal University of *Rio Grande do Sul*, Brazil, Federal University of *Caxias do Sul*, Brazil, and Oregon Graduate Institute, USA. The corpus has been distributed by LDC (LDC2006S16).

The utterances consist of both read speech (for phonetic coverage) and responses to questions (for spontaneous speech) from a variety of regions in Brazil. The acoustic environment was not controlled, in order to allow for background conditions that would occur in application environments.

Although useful, the Spoltech corpus has several problems. Some audio files do not have their corresponding orthographic and phonetic transcriptions, and vice versa. Another problematic aspect is that both phonetic and orthographic transcriptions have many errors. So a major revision and correction of multiple files was made. In this research, the corpus was composed by 477 speakers, which corresponds to 4.3 hours of audio. The speech signal was resampled from 44,100 Hz to 22,050 Hz with 16 bits.

3.6 West Point corpus

The West Point Brazilian Portuguese Speech corpus [48] was created by the USA government and has been distributed by LDC (LDC2008S04). The utterances consist of read speech (296 phrases) and was composed by 60 male and 68 female, native and nonnative speakers.

The West Point corpus also has audio files that do not have their corresponding orthographic and phonetic transcriptions. Other problematic aspect is the existence of

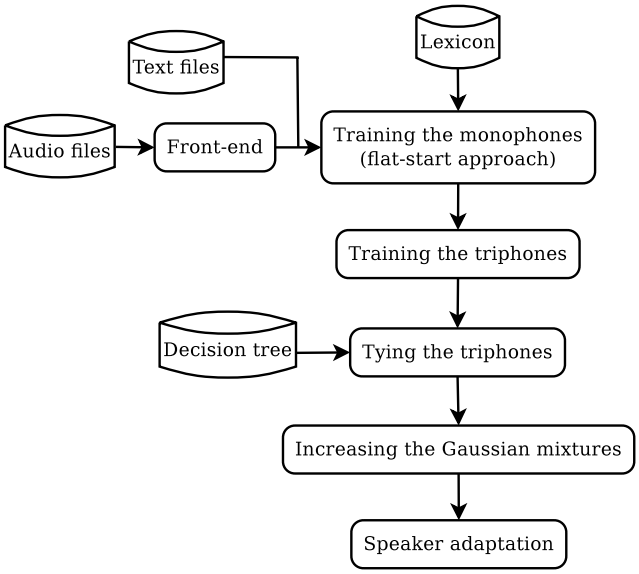


Fig. 3 The acoustic model development process

records with faults such as noise and unclear speech. Thus, a preprocessing stage was performed, and 7,920 audio files with native speakers were selected, which corresponds to 8 hours of audio. The speech signal was resampled from 22,050 Hz to 11,025 Hz with 16 bits.

4 An acoustic model for BP

This section describes the development of an acoustic model for BP (see Fig. 3). 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 digitized voice, transcribed at the level of words (orthography) and/or at the level of phones. In the sequel, some aspects of developing a model for BP are discussed.

The current version of the Julius decoder works only with MFCC front ends, and, therefore, MFCC was adopted for convenience. More specifically, the front end consists of the widely used 12 MFCCs [49] using C0 as the energy component, and computed every 10 milliseconds (i.e., 10 ms is the frame shift) for a frame of 25 ms. These static coefficients are augmented with their first and second derivatives to compose a 39-dimensional parameter vector per frame. Finally, the cepstral mean subtraction technique was used to normalize the MFCCs coefficients [37].

The acoustic models were iteratively refined [50]. A flat-start approach was adopted, starting with continuous single-component mixture monophone models, the HMMs were gradually improved to finally have mixtures of multiple Gaussians to model the output distributions. The set of HMMs was composed by tied-state triphones. During all the training process, the embedded Baum–Welch algorithm [51] was used to re-estimate the models.

The initial acoustic models for the 39 phones (38 monophones and a silence model) used 3-state left-right HMMs. The silence model was trained and then copied to create the tied short-pause (*sp*) model with only one acoustic state. The sp has a direct transition from the entry to the exit state. After that, *cross-word* triphone models were built from the monophone models. Transition matrices of triphones that share the same base phone were tied.

Given a set of categories (also called *questions*), a decision tree specific for BP was designed for tying the triphones with similar phonetic characteristics. To illustrate, some vowels and consonants classification rules used to build the decision tree are listed below:

```
...
QS "R_V-Close"   *+i,*+e,*+o,*+u
QS "R_V-Front"   *+i,*+E,*+e
QS "R_Palate"    *+S,*+Z,*+L,*+J
QS "L_V-Back"    u-*,o-*,O-*
QS "L_V-Open"    a-*,E-*,O-*
...
```

Notice that for a triphone system, it is necessary to include questions referring to both the right and left contexts of a phone. The questions should progress from wide, general classifications (such as consonant, vowel, nasal, diphthong, etc.) to specific instances of each phone. Ideally, the full set of questions loaded using the HTK *QS* command would include every possible context that can influence the acoustic realization of a phone and can include any linguistic or phonetic classification that may be relevant.

After tying, the number of component mixture distributions was gradually increased up to 14 Gaussians per mixture to complete the training process. The scripts used for developing the acoustic model and the decision tree, specifics for BP, were made available [17].

5 A language model for BP

Training the language model requires the chosen dictionary (lexicon) and a file with the sentences from which the words' counts will be extracted [52]. The dictionary is required because the words found in the training sentences that are not in the dictionary will not be counted. Figure 4 illustrates the general form of the language model training and test processes.

The *n*-gram models are straightforward to construct except for the issue of smoothing, a technique used to better estimate probabilities when there is insufficient data to accurately estimate them. An enormous number of techniques have been proposed for smoothing *n*-gram models [53]. The Kneser–Ney smoothing technique was used in this work. It is an extension of the absolute discounting algorithm and

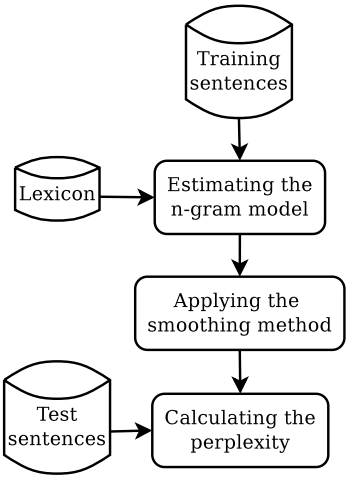


Fig. 4 The language model development process

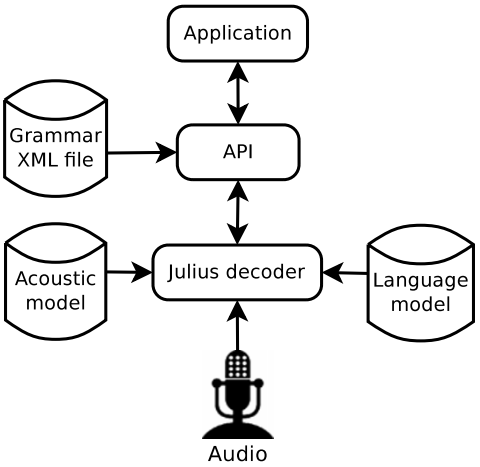


Fig. 5 The designed API for easing the task of driving the Julius speech recognizer

adopts the heuristic that a unigram probability should not be proportional to the number of occurrences of a word, but instead to the number of different words (contexts) that it follows [54].

A more detailed description of the language models developing process can be seen in Sect. 7.2. In the sequel, an API is proposed in order to facilitate using the high-performance Julius speech decoder.

6 An application programming interface for the recognizer

While trying to promote the widespread development of applications based on speech recognition, the authors noted that it was not enough to make available resources such as language models. These resources are useful for speech scientists, but most programmers demand an easy-to-use API. Hence, it was necessary to complement the documentation and code that is part of the Julius package [39].

The recent version of the Julius decoder is fully SAPI 5.1 compliant, but it assumes that the language is Japanese. It is troublesome to use SAPI with Julius in the case of other languages, such as Portuguese. Hence, Julius does not support neither SAPI nor the JSAPI recognition specifications, but it has its own API, which is for C/C++ programming.

As explained, the current work aims flexibility with respect to the programming language. Besides, the goal is to support Windows, Linux, and potentially other operating systems. The solution was to propose a simple API, with the required functionality to control Julius, and with implementations for C++ on Linux and the .NET platform on Windows. Another implementation of the API, targeting Java programmers, is under development. The support for Windows is based on the adoption of the Common Language Runtime specification, which enables communication between the languages supported by the .NET platform (C#, Visual Basic, J#, and others).

Table 3 Main API methods and events

Methods and events	Basic description
SREngine	Engine's setup and control
loadGrammar	Load a grammar file
startRecognition	Start the recognition process
stopRecognition	Stop the recognition process
OnRecognition	Receive the recognition results
OnSpeechReady	Indicate that the engine is active

The proposed API allows the real-time control of the Julius ASR engine and the audio interface. As shown in Fig. 5, the applications interact with the Julius decoder through the API.

Since the API supports the component object model automation, it is possible to access and manipulate (i.e., set properties of or call methods on) shared automation objects that are exported by other applications. From a programming point of view, the API consists of a main class referred to as *SREngine*. This class exposes to the applications a set of methods and events that are described in Table 3.

The *SREngine* class enables applications to control aspects of the Julius decoder. The application can load the acoustic and language models to be used, start and stop recognition, and receive events and recognition results.

The *loadGrammar* method loads a context-free grammar[2] file specified in SAPI XML format. To make this pos-

[2]The context-free grammar acts as the language model. It provides the recognizer with rules that define what the user is expected to say.

sible, a flexible converter was developed by the authors. This toolkit allows users to convert a recognition grammar specified in XML, according to the SAPI Text Grammar Format [55], into the Julius format.[3] The conversion procedure uses the SAPI grammar rules to find the allowed connection of words, using word category names as terminal symbols. It also defines word candidates in each category, with their pronunciation information. It should be noted that the converter does not support recursive rules in the grammar, a feature that is supported by Julius.

The *startRecognition* method, responsible for starting recognition, activates the grammar rules and opens the audio stream. Similarly, the *stopRecognition* method deactivates the rules and closes the audio stream.

In addition to the methods, some events treatment is also supported. The *OnSpeechReady* event signals that the engine is active to recognize. In other words, it occurs whenever the *startRecognition* method is invoked. Now the *OnSRecognition* event occurs whenever a recognition result is available with an associated confidence measure. A confidence measure of the recognition results is essential to real applications because there are always recognition errors and therefore the recognition results need to be accepted or rejected. The utterance and confidence score are passed from the API to the application through the *RecoResult* class.

```
namespace Test {
  public partial class Form1 : Form {
    private SREngine engine = null;
    public Form1() {
      SREngine.OnRecognition += handleResult;
    }
    public void
    handleResult(RecoResult result) {
      Console.Writeline(result.
      getConfidence() + " | "
        + result.getUterrance() + "\n");
    }
    private void
    but_Click(object sender, EventArgs e) {
      engine=new
      SREngine(@".\LaPSAM\LaPSAM1.5.jconf");
      engine.loadGrammar(@".
      \Gramatica\grammar.xml");
      engine.startRecognition();
    }
  }
}
```

With the limited set of methods and events presented above it is easy to build compact speech recognition applications using the Julius decoder. Listing above presents a sample code that recognizes from a context-free XML grammar and shows the results on screen. The API resources like source codes, libraries, and sample applications are publicly available [56].

Having presented a summary of the most important developed resources for BP, the next section discusses some results achieved.

7 Experimental results

This section presents the baseline results. The first experiment evaluates the G2P converter effectiveness and the influence of the phonetic dictionary in a speech recognition system performance. All the tests were executed on a computer with Core 2 Duo Intel processor (E6420 2.13 GHz) and 1 GB of RAM.

7.1 Evaluation of the G2P converter and corresponding dictionary

Three phonetic dictionaries were compared. The first one was based on the G2P module with the proposed rules described in Sect. 3. The second dictionary used the original G2P described in [42]. The third dictionary followed the approach described in [57], which is based on a machine learning approach using a decision tree. The third dictionary [57] adopts a phonetic alphabet with only 34 phones, while the other two use 38. This reflects in the acoustic modeling, with the third dictionary having 34 HMMs, while the others were tested with the same acoustic model (with 38 HMMs). The three dictionaries were chosen because they represent the evolution of our research in this topic, with the third dictionary being the first attempt, which was followed by implementing the proposed in [42] and improving it.

If there is a labeled dataset with the correct transcriptions, it can be used for tests and play the role of an "oracle" that provides the right answer. However, this approach can be questioned with respect to the correctness of the labeled dataset. Alternatively, this work assesses the dictionaries by observing their impact on WER, which is the adopted figure of merit for the ASR systems.

The acoustic models were built using only the West Point corpus, which was divided into two disjoint data sets for training and test. In the experiments, the West Point training set was composed by 6,334 files, corresponding to 384 minutes, and the test set used the remaining 1,586 files corresponding to 96 minutes.

The vocabulary was the 679 words present in the West Point transcriptions. Bigram language models were used, with the number of sentences for training varied from 1,000 to 180,000. One thousand disjoint sentences were used to measure the perplexity of each configuration. The sentences

[3] Julius supports both *n*-gram and grammars for command-and-control applications.

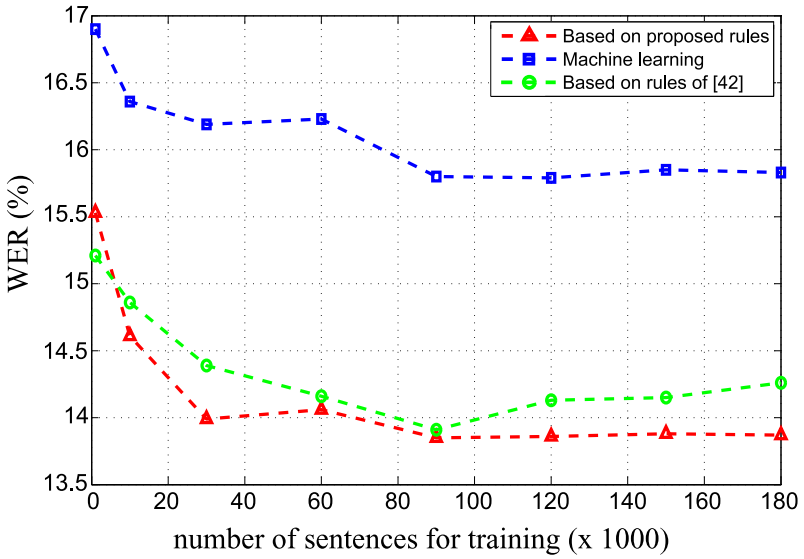


Fig. 6 A comparison among two dictionaries based on rules and one based on machine learning [57]

Table 4 Evaluating the language model perplexity against the number of sentences used to train it

	Number of sentences							
	1 k	10 k	30 k	60 k	90 k	120 k	150 k	180 k
PP	50	33.3	26.2	21.7	19.2	16.4	15.7	14.8

were extracted solely from the CETENFolha corpus. It is important to observe that, in terms of sentences, the West Point transcriptions and the CETENFolha corpus can be considered disjoint. Table 4 shows the perplexities found in these experiments. As expected, the perplexity diminishes as the number of sentences used in the training increases [9].

Because there was no interest in real-time operation, the HDecode decoder was used to perform the tests and the results are shown in Fig. 6, with the WER decreasing as the number of sentences used to train the language model increases.

On the executed experiments, the results remained nearly constant in some intervals. The reason for that could be related to a saturation of the language model, when almost all common n-gram sequences already appear in the language model, but rare ones are still unlikely to be seen in the training corpus. However, these uncommon n-grams are the ones whose probability is the hardest to estimate correctly, so adding small quantities of new data does not correspondingly improve the language model.

The comparison between the different approaches should consider as well the size of the resulting transducers and other properties which may also be quite relevant [40], such as the fact that the machine learning approach requires lexical alignment, whereas the rule-based approach does not.

As expected, best results were obtained with the rule-based approach, but one should take into account the fact that the machine learning one was trained with a hand-labeled pronunciation dictionary [57]. Furthermore, the suggested changes to the set of rules described in [42] consistently improved the performance of the ASR system.

7.2 Evaluation of the overall ASR system

While the previous results were obtained with a simplified setup, the next ones were obtained with all the developed resources. The acoustic model was initially trained using only the LapsStory corpus and the UFPAdic. After that, the HTK software was used to adapt the acoustic model, using the MLLR and MAP techniques with the Spoltech corpus, according to the steps described in [37]. This adaptation process was used to combat acoustic mismatches and is described in [45]. Both MAP and MLLR were used in the supervised training (offline) mode. The LapsBenchmark corpus was used to evaluate the systems.

Several language models were tested, and the results are shown in Table 5. The first language model (LM_1) was designed solely with the LapsNews text corpus, the second model (LM_2) with sentences extracted from CETENFolha, Spoltech, LapsStory, West Point, and OGI22 [58] corpora, and the last one (LM_3) was a combination of previous models, in other words, it encompasses the phrases present in LM_1 and LM_2.

The sentences used to measure the perplexity of each configuration were ten thousand sentences extracted from the CETENFolha corpus, unseen during the training phase. All the language models were designed with Kneser–Ney smoothing. The WER was also evaluated for the language models using the Julius decoder. Julius decodes using two passes (forward and backward search) with bigram and trigram language models on the first and second passes, respectively.

The LM_2 and LM_3 achieved equivalent results, and, for the next experiments, the LM_3 language model was adopted, and the performance measures were the WER and xRT. The scripts used to build and apply these statistical language models are publicly available [17].

The pruning process is implemented at each time step by keeping a record of the best hypotheses overall and deactivating 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 Figs. 7 and 8. The Julius decoding parameters as well as the beam width can be adjusted for the respective passes. The beam width value was varied on the first pass and was set to 200 on the second pass.

It was observed 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.

An extra comparison was made with the commercial software IBM ViaVoice. The evaluation process was carried out in two stages, speaker-independent and speaker-dependent models. The results are shown in Table 6. The decoding parameters were optimized. The beam width value was set to 220 and 2,000 for HDecode and Julius, respectively. It was necessary to keep the xRT factor value around one. In the case of IBM ViaVoice, the acoustic and language models provided by the software were used to perform the experiments. The LapsBenchmark corpus was again used to evaluate the systems.

The IBM ViaVoice requires a session of speaker adaptation, which had to be tricked in order to mimic a speaker-independent operation. Hence, for the first stage, the speaker adaptation process for ViaVoice was carried out using the voice of six speakers, 3 men and 3 women, which corresponds to 10 minutes of audio. The xRT could not be mea-

Table 5 Comparing the language models obtained with different text corpora and tested with the same acoustic data

Corpus	LM_1	LM_2	LM_3
Phrases	672,718	1,526,030	2,034,611
Distinct words	151,858	214,717	271,633
Bigram PP	482	240	246
Trigram PP	385	143	145
WER(%)	42.21	29.03	29.57

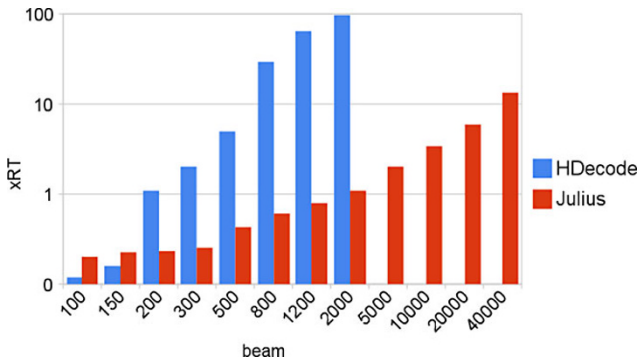
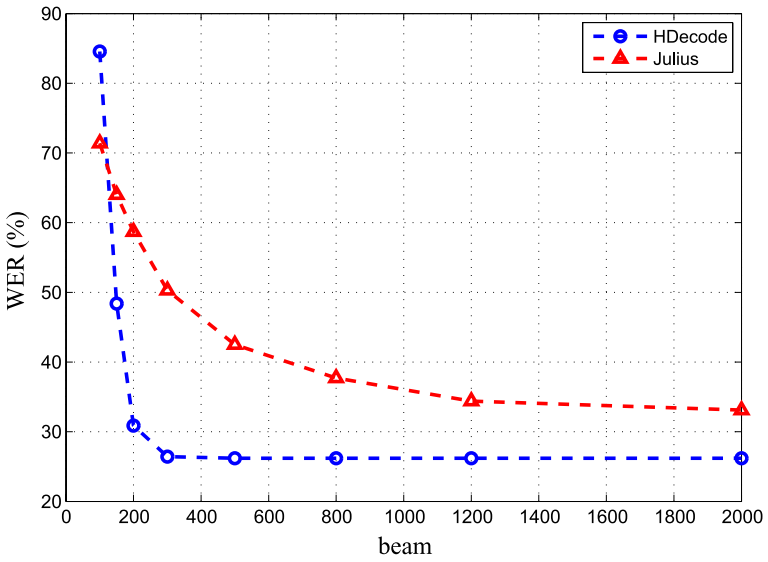


Fig. 7 Variation of xRT with the beam width parameter

Table 6 Systems comparison using speaker independent and dependent models

Decoder	Independent models		Dependent models	
	WER(%)	xRT	WER(%)	xRT
Julius	29.00	0.99	13.30	0.99
HDecode	20.13	1.20	6.81	0.91
ViaVoice	29.30	–	17.30	–

Fig. 8 Variation of WER(%) with the beam width parameter



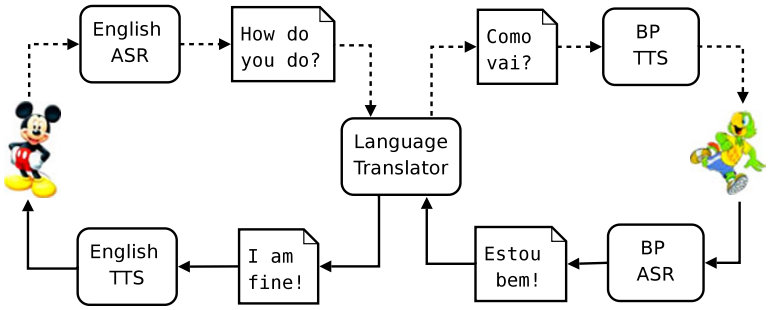


Fig. 9 Block diagram of the developed translation system for spoken dialogs

sured for ViaVoice due to the adopted procedure for invoking the recognizer in batch. Note that Julius and ViaVoice had almost the same performance, while both were outperformed by HDecode.

Two male speakers were used in the speaker-dependent evaluation, with each one contributing with 10 minutes of his voice. The MLLR and MAP adaptation techniques were again used for the models adopted in Julius and HDecode. In the case of IBM ViaVoice, the standard adaptation process (guided by the software) was adopted. As expected, the speaker adaptation increased the performance of all decoders and allowed Julius to outperform ViaVoice. HDecode achieved again the best result.

The speaker independent acoustic model and the LM_3 language model are publicly available [17].

8 Applications of the developed system and resources

While the previous section reported numerical results, this one describes applications of the developed system and resources given that this work aims at reaching external technology adopters.

8.1 SpeechOO

The public resources for BP developed by the FalaBrasil project have been used by the open-source community to develop speech-enabled applications such as the SpeechOO, a dictation pad for OpenOffice.org [59]. It is a speech recognition extension that can get utterances from Julius decoder and append it to the current *Writer* tool document.

The current version of SpeechOO is available [60] and works only under GNU/Linux systems. The prototype was developed in the Java programming language and uses Java Native Interface (JNI) to communicate with the proposed API. It proves the concept of running mixed extension: Java plus C++ wrapped with JNI.

The SpeechOO project is maintained by members of the FalaBrasil project and the Computer Science Department at *São Paulo* University (USP), Brazil.

8.2 Speech-to-speech machine translation

A second application was developed at the Federal University of *Pará* (UFPA), Brazil, to illustrate the interface between speech processing and natural language processing (NLP) in the context of statistical machine translation (SMT) [61]. The goal of the developed system is to allow a spoken dialog between native speakers of English and BP. The current version has limitations, but it is operational. The system will be fully described in another work, and it is discussed here to exemplify the advantages of having publicly available ASR and TTS systems for BP.

As depicted in Fig. 9, a speaker can say a phrase in BP, and the BP ASR translates it to text, which is then converted to English by the SMT module. This text in English is the input of an English TTS. A similar process is used in the conversion of phrase spoken in English. The BP ASR is the one described in this work with the acoustic model adapted with 10 minutes of audio from the target speaker, the LM_3 language model, and the Julius decoder. The BP TTS is the one presented in [19]. The English ASR system was built using the free speech corpus and acoustic model available in the VoxForge repository [62], and the Julius rev.4.1.5 decoder. The English TTS was the open-source FreeTTS 1.2 system [63].

The preparation of the data is an essential stage of any developing SMT system. Two different data sets are required, training and test. These data sets have to be provided as aligned sentences (one sentence per line), in two files, one for the BP sentences and one for the English sentences. In this task, we used part of the PAR-C parallel corpus [64].

Although useful, the PAR-C corpus presented some phrase alignment problems. Thus, a manual revision stage was performed, and 881 pairs of BP-English parallel sentences were selected. It is known that the number of pairs of sentences is considered small by comparing results with other SMT researches [65]. An improved SMT system can be potentially designed when using all the PAR-C corpus, but the goal here is to validate the interfaces among speech and SMT tools.

The final training corpus for the SMT prototype was composed of 45,245 simple tokens (21,656 in BP and 23,589

Table 7 Examples of source, recognized, and translated sentences

Source sentence	Recognized sentence	Translated sentence
eles estão colocando armadilhas nas fazendas onde já ocorreram os ataques	eles estão colocando armadilhas nas fazendas onde já ocorreram os ataques	they are by placing traps in farms where already there were the ataques
somente umas trezentas e vinte foram inauguradas em território americano	somente dois trezentos e vinte foram inauguradas em território americano	somente two hundred and twenty were opened in u.s. territory
a secretaria estadual de saúde distribuirá cem mil preservativos no carnaval	a secretaria estadual de saúde distribuirá cem mil preservativos no carnaval	the basic state health distribuirá 100 thousand condons in carnival
a maioria dos passageiros do barco naufragado era de crianças	a maioria dos passageiros do barco naufragado era de crianças	the majority of passageiros of boat wrecked was children
em florianópolis foi registrado dois graus celsius na manhã de domingo	em florianópolis foi registrado dois graus é ou se os na manhã de domingo	in florianópolis was recorded two graus is or if the on sunday morning
se for eleito vocês vão ver o meu trabalho	se for eleito vou ser ou ver o meu trabalho	if becomes eleito am be or see my work

in English) and 793 pairs of BP-English parallel sentences. Now the test corpus was composed of 88 pairs of BP-English parallel sentences with 4,269 tokens (2,052 in BP and 2,217 in English), unseen during the training phase.

This work used the Moses system [66]. Moses is an open-source toolkit for SMT and uses standard external tools such as the SRILM for language modeling and the GIZA++ for word alignments [67].

8.2.1 Training

SMT training is based on building two statistical models, a language model and a translation model [68]. These models are built from a training parallel corpora and calculate the probability of a given source phrase to be translated to a target phrase.

The first step in the training process consisted of tokenizing the training corpus (separation of minimum processing units: words, punctuation characters). The last step of preprocessing the corpus is responsible for converting the training data to lowercase.

During the translation process, the language model is used to order the sentences generated automatically according to their probability of being *correct* sentences. The full 881 tokenized sentences were used to build the language models. The trigram language models were designed with Kneser–Ney smoothing.

Finally, the training is completed with the generation of the phrase translation table. The phrase-table lists the sentences according to their translation likelihood and outputs the phrase and reordering tables needed for decoding. A training script provided by the Moses' toolkit was used to produce the phrase-table. The phrase-table produced from training contains a total of 72,884 phrases for BP to English and 73,858 phrases for English to BP.

8.2.2 Evaluation

The statistical language and translation models were evaluated using the test corpus, and the performance presented by the developed system was analyzed in respect of the bilingual evaluation understudy (BLEU) and the National Institute of Standards and Technology (NIST) measures [69].

The developed translator presented a BLEU value of 0.0849 and a NIST value of 3.9124 for BP to English and a BLEU value of 0.0839 and a NIST value of 3.8291 for English to BP. As mentioned, the SMT can be improved following, e.g., the guidelines in [68, 70]. Some interesting examples are shown in Table 7. It could be observed that errors of the ASR module significantly decreased the performance of the SMT module, given that the recognized text sentences may be nonsense.

9 Conclusions

As discussed, one of the biggest problems in building LVCSR systems is the lack of data for training and testing. Shared databases between North American and European researchers had been one of the main reasons for the progresses achieved on the last decades in these regions [9]. For the BP, however, there are not common databases. This paper presented an LVCSR system for BP and the corresponding results. All the developed modules can be obtained at the FalaBrasil project repository [17]. In summary, the specific tools and resources for BP that were released are:

- A phonetic dictionary with 65,532 words.
- A rule-based G2P converter (executable file).
- A newspaper text corpus with nearly 672 thousand formatted sentences.
- Two multiple speakers audio corpora corresponding together to approximately 16.5 hours of audio. Only part of

these corpora is publicly available, which corresponds to 9 hours and 54 minutes of audio.
– Software recipes to design statistical acoustic and language models.
– A speaker-independent HTK format acoustic model and a trigram ARPA format language model.
– An open-source API in order to control the Julius speech decoder. The API contains its own SAPI XML to Julius grammar converter.

In fact, after making available resources, just recently the interest on them significantly increased, due partially by the consistent and easy-to-use proposed API. The API allows one to abstract most of the details and achieves seamless integration of the Julius recognizer with popular programming languages.

Although the Julius decoder presented the worst performance in all conducted tests, it has been shown that it can eventually outperform HDecode [71]. Currently, the authors have not been able to make Julius achieve the same level of performance of HDecode in the available BP data. In spite of that, Julius has a flexible license that allows its commercial use, which is important for users that intend to develop products.

Besides presenting state of art results for BP, this work described the successful actions to promote the development of speech-enabled softwares for BP; for instance, the SpeechOO project, which is a voice recognition extension for OpenOffice.org. Another example of application is the human–robot project conducted by the mechatronics engineering students from the *Brasília* University (UnB), Brazil, which has also been using the described speech recognition resources to control a Pioneer robot via gestures and voice commands [72].

Future works include expanding both the audio and text corpora, 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 applications such as the SMT-based dialog translator will also help disseminating the technology.

Acknowledgements This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, project no. 482148/2009-8, Fundação para a Ciência e a Tecnologia (FCT), Portugal, project no. PTDC/PLP/72404/2006, and Fundação de Amparo à Pesquisa do Estado do Pará.

References

1. Rabiner L, Juang B (1993) Fundamentals of speech recognition. PTR Prentice Hall, Englewood Cliffs
2. Huang X, Acero A, Hon H (2001) Spoken language processing. Prentice-Hall, New York
3. Dutoit T (2001) An introduction to text-to-speech synthesis. Kluwer Academic, Dordrecht
4. Taylor P (2009) Text-to-speech synthesis. Cambridge University Press, Cambridge
5. Allen J, Hunnicutt MS, Klatt DH, Armstrong RC, Pisoni DB (1987) From text to speech: the MITalk system. Cambridge University Press, Cambridge
6. Odell J, Mukerjee K (2007) Architecture, user interface, and enabling technology in Windows Vista's speech systems. IEEE Trans Comput 56(9):1156–1168
7. www.google.com/chrome. Visited in June 2010
8. Schramm M, Freitas L, Zanuz A, Barone D (2000) A Brazilian Portuguese language corpus development. In: International conference on spoken language processing, vol 2, pp 579–582
9. Teruszkin R, Vianna F (2006) Implementation of a large vocabulary continuous speech recognition system for Brazilian Portuguese. J Commun Inf Syst 21:204–218
10. Ynoguti CA, Violaro F (2008) A Brazilian Portuguese speech database. In: XXVI simpósio Brasileiro de telecomuniçacões
11. Neto J, Meinedo H, Viveiros M, Cassaca R, Martins C, Caseiro D (2008) Broadcast news subtitling system in Portuguese. In: IEEE international conference on acoustics, speech, and signal processing
12. Neto J, Martins C, Meinedo H, Almeida L (1997) The design of a large vocabulary speech corpus for Portuguese. In: Proceedings of the European conference on speech technology
13. Paul D, Baker J (1992) The design for the Wall Street Journal-based CSR corpus. In: Proceedings of the international conference on spoken language processing
14. Ribeiro ITM, Duarte I, Matos G (1998) Corpus de diálogo CORAL. In: III encontro para o processamento computacional da língua Portuguesa escrita e Falada
15. Trancoso I, Martins R, Moniz H, Silva A, Ribeiro M (2008) The LECTRA corpus: Classroom lecture transcriptions in European Portuguese. In: Language resources and evaluation conference
16. Valtchev V, Odell JJ, Woodland PC, Young SJ (1997) MMIE training of large vocabulary recognition systems. Speech Commun 22(4):303–314
17. www.laps.ufpa.br/falabrasil. Visited in June 2010
18. Vandewalle P, Kovacevic J, Vetterli M (2009) Reproducible research in signal processing—what, why, and how. IEEE Signal Process Mag 26:37–47
19. Couto I, Neto N, Tadaiesky V, Klautau A, Maia R (2010) An open source HMM-based text-to-speech system for Brazilian Portuguese. In: 7th international telecommunications symposium
20. www.microsoft.com/speech/. Visited in June 2010
21. Neto N, Sousa E, Macedo V, Adami A, Klautau A (2005) Desenvolvimento de software livre usando reconhecimento e síntese de voz: O estado da arte para o Português Brasileiro. In: 6th forum internacional software livre
22. www.microsoft.com/portugal/mldc/downloads.mspx. Visited in June 2010
23. Santos S, Alcaim A (2002) Um sistema de reconhecimento de voz contínua dependente da tarefa em língua portuguesa. Rev Soc Brasil Telecomun 17(2):135–147
24. Fagundes R, Sanches I (2003) Uma nova abordagem foneticofonologica em sistemas de reconhecimento de fala espontinea. Rev Soc Brasil Telecomun 95:225–239
25. Silva E, Baptista L, Fernandes H, Klautau A (2005) Desenvolvimento de um sistema de reconhecimento automático de voz contínua com grande vocabulário para o Português Brasileiro. In: XXV congresso da sociedade Brasileira de computação
26. Abad A, Trancoso I, Neto N, Ribeiro M (2009) Porting an European Portuguese broadcast news recognition system to Brazilian Portuguese. In: Interspeech, Brighton, UK

27. Rabiner L (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77(2):257–86
28. Juang H, Rabiner R (1991) Hidden Markov models for speech recognition. Technometrics 33:251–272
29. Deshmukh N, Ganapathiraju A, Picone J (1999) Hierarchical search for large-vocabulary conversational speech recognition. IEEE Signal Process Mag 84–107
30. Jevtić N, Klautau A, Orlitsky A (2001) Estimated rank pruning and Java-based speech recognition. In: Automatic speech recognition and understanding workshop
31. Ladefoged P (2001) A course in phonetics, 4th edn. Harcourt Brace, New York
32. www.phon.ucl.ac.uk/home/sampa/. Visited in June 2010
33. Antoniol G, Fiutem R, Flor R, Lazzari G (1993) Radiological reporting based on voice recognition. In: Human–computer interaction. Lecture notes in computer science, vol 753. Springer, Berlin, pp 242–253
34. Lee CH, Gauvain JL (1993) Speaker adaptation based on MAP estimation of HMM parameters. In: IEEE ICASSP, pp 558–561
35. Ralf SG, Kompe R (2000) A combined MAP + MLLR approach for speaker adaptation. Proc Sony Res Forum 9:9–14
36. Bellman R (1957) Dynamic programming. Princeton University Press, Princeton
37. Young S, Ollason D, Valtchev V, Woodland P (2006) The HTK book, version 3.4. Cambridge University Engineering Department, Cambridge
38. Stolcke A (2002) SRILM—an extensible language modeling toolkit. In: International conference on spoken language processing
39. Lee A (2009) The Julius book, 0.0.2 ed., rev 4.1.2
40. Caseiro D, Trancoso I, Oliveira L, Viana C (2002) Grapheme-to-phone using finite-state transducers. In: In IEEE workshop on speech synthesis
41. Teixeira A, Oliveira C, Moutinho L (2006) On the use of machine learning and syllable information in European Portuguese grapheme-phone conversion. In: Computational processing of the Portuguese language. Lecture notes in computer science, vol 3960. Springer, Berlin, pp 212–215
42. Silva D, de Lima A, Maia R, Braga D, de Moraes JF, de Moraes JA, Resende F Jr (2006) A rule-based grapheme-phone converter and stress determination for Brazilian Portuguese natural language processing. In: VI international telecommunications symposium
43. Faria A (2003) Applied phonetics: Portuguese text-to-speech. Tech Rep. University of California
44. acdc.linguateca.pt/cetenfolha/. Visited in June 2010
45. 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. In: XXVII simpósio Brasileiro de telecomuniçacões
46. Cirigliano RJ, Monteiro C, de F Barbosa FL, Resende FL Jr, Couto L, Moraes J (2005) Um conjunto de 1000 frases foneticamente balanceadas para o Português Brasileiro obtido utilizando a abordagem de algoritmos genéticos. In: XXII simpósio Brasileiro de telecomuniçacões
47. Neto N, Silva P, Klautau A, Adami A (2008) Spoltech and OGI-22 baseline systems for speech recognition in Brazilian Portuguese. In: International conference on computational processing of Portuguese language—PROPOR
48. Weimar F, Barone D, Adami A (2010) A baseline system for continuous speech recognition of Brazilian Portuguese using the West Point Brazilian Portuguese speech corpus. In: International conference on computational processing of Portuguese language
49. Davis S, Merlmestein P (1980) Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans ASSP 357–366
50. Woodland P, Young S (1993) The HTK tied-state continuous speech recognizer. In: Proceedings of the Eurospeech'93, Berlim
51. Welch LR (2003) Hidden Markov models and the Baum–Welch algorithm. IEEE Inf Theory Soc Newslett 53:10–12
52. Silva E, Pantoja M, Celidnio J, Klautau A (2004) Modelos de linguagem n-grama para reconhecimento de voz com grande vocabulários. In: III workshop em tecnologia da informação e da linguagem humana
53. Chen SF, Goodman J (1999) An empirical study of smoothing techniques for language modeling. Comput Speech Lang 13:359–394
54. Kneser R, Ney H (1995) Improved backing-off for M-gram language modeling. In: IEEE international conference on acoustics, speech and signal processing, pp 181–184
55. msdn.microsoft.com/en-us/ms723632(VS.85).aspx. Visited in June 2010
56. code.google.com/p/lapsapi/. Visited in June 2010
57. Hosn C, Baptista LAN, Imbiriba T, Klautau A (2006) New resources for Brazilian Portuguese: results for grapheme-to-phoneme and phone classification. In: VI international telecommunications symposium
58. Lander T, Cole R, Oshika B, Noel M (1995) The OGI 22 language telephone speech corpus. In: Proceedings of the Eurospeech, Madrid
59. Colen WD, Batista P (2010) Veja mampe, sem as mpos! SpeechOO, uma extenspo de ditado para o BrOffice.org. In: 11th fórum internacional software livre
60. code.google.com/p/speechoo/. Visited in June 2010
61. Koehn P, Och FJ, Marcu D (2003) Statistical phrase-based translation. In: Proceedings of the human language technology, pp 127–133
62. http://www.voxforge.org/. Visited in June 2010
63. http://freetts.sourceforge.net. Visited in June 2010
64. nilc.icmc.usp.br/lacioweb/parc.php. Visited in June 2010
65. Aziz W, Pardo T, Paraboni I (2009) Fine-tuning in Portuguese–English statistical machine translation. In: 7th Brazilian symposium in information and human language technology
66. Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N, Cowan B, Shen W, Moran C, Zens R, Dyer C, Bojar O, Constantin A, Herbst E (2007) Moses: open source toolkit for statistical machine translation. In: Proceedings of association for computational linguistics, pp 177–180
67. Och FJ, Ney H (2000) Improved statistical alignment models. In: Proceedings of association for computational linguistics, pp 440–447
68. Caseli H, Nunes I (2009) Statistical machine translation: little changes big impacts. In: 7th Brazilian symposium in information and human language technology, pp 1–9
69. Zhang Y, Vogel S, Waibel A (2004) Interpreting BLEU/NIST scores: How much improvement do we need to have a better system. In: 4th international conference on language resources and evaluation
70. Koehn P, Hoang H (2007) Factored translation models. In: Empirical methods on natural language processing, pp 868–876
71. Yao X, Bhutada P, Georgila K, Sagae K, Artstein R, Traum D (2010) Practical evaluation of speech recognizers for virtual human dialogue systems. In: 7th international language resources and evaluation
72. groups.google.com.br/group/human-robot-interaction/. Visited in June 2010