Sociedade de Engenharia de Áudio

Artigo de Congresso

Apresentado no 15º Congresso de Engenharia de Áudio da AES Brasil
23 a 25 de Outubro de 2017, Florianópolis, SC

Assessing the Performance of a Post-Processed End-to-End Speech Recognizer for Brazilian Portuguese Using Deep Neural Networks

Igor M Quintanilha,[1] Sergio L. Netto,[1] and Luiz W. P. Biscainho[1]

[1] Universidade Federal do Rio de Janeiro, DEL/Poli & PEE/COPPE
Rio de Janeiro, RJ, 21941-598, Brasil

igor.quintanilha@smt.ufrj.br, sergioln@smt.ufrj.br, wagner@smt.ufrj.br

ABSTRACT

In a previous approach to the development of a character-level end-to-end automatic speech recognition (ASR) system using deep learning, most of the network mistakes seemed easy to identify. This paper investigates the effects of two different post-processing schemes towards the automatic correction of such errors without resorting to any complex decoding phase: a simple spell-checker algorithm, which produced no noticeable improvement on the character error rate; and a grapheme-to-phoneme converter, which confirmed that almost 5% of the errors found are phonetically motivated, and thus amenable to be mitigated by a properly designed language model.

0 INTRODUCTION

In the last decade or so, the advent of deep learning (DL) has drastically impacted areas [1] so diverse as pedestrian detection [2], face recognition [3], disease identification [4], machine translation [5], image classification [6], detection [7] and segmentation [8].

In automatic speech recognition (ASR), for instance, a major accuracy improvement was achieved in 2012, when the well-established Gaussian mixture model (GMM) and hidden Markov model (HMM) were replaced by a deep neural network (DNN) [9]. From that work on, the ASR area has seen successive benchmark breakings [10, 11, 12], even surpassing the human capability in some current scenarios [13].

Unfortunately, building an entire ASR system is still a complex task, requiring a GMM module to obtain the initial frame-level labels, multiple stages of training with hand-designed features, and an expert to determine the optimal configuration of many hyperparameters. Recently, following the trends in the computer vision area—where end-to-end systems have indeed transposed human capabilities—, researchers have been trying to find a neural-based system to perform speech recognition in an end-to-end manner. Among

such end-to-end ASR systems, two have excelled: the connectionist temporal classification (CTC) [14], and the sequence-to-sequence models (seq2seq) with attention mechanism [15].

The CTC method, proposed by Graves *et al.* [14], was the first successful end-to-end ASR algorithm, and since then has been widely employed by giant companies [16, 12] to develop their own systems. This simple method overcomes one of the most challenging problems in ASR, which is the time alignment between the frame-level features and the final text transcription, thus being able to transcribe an utterance without knowing precisely where it occurs.

Another effective algorithm is the seq2seq with attention mechanism [17], which first appeared in neural machine translation, where an encoder compresses the input language sequence into a vector, and a decoder (supervised by the attention mechanism) decompresses this vector into the desired language.

This paper focuses on the analysis of a CTC-based end-to-end ASR for the Portuguese language. Previous work [18, 19] has shown that gathering several small Portuguese datasets enabled the training of a character-level ASR deep model, yielding a character error rate (CER) of 23.15%, only 8.4%–12.9% below commercial systems, without using a language model or any additional decoding scheme. In such a system, most of the resulting errors were due to homophone or near-homophone transcriptions (*e.g.*, 'dessert' and 'desert'), the same conclusion the authors of [20] had arrived at.

In this work, we investigate the impact of two post-processing methods on the CER attained by the end-to-end ASR system originally proposed in [18]. Specifically, a spell checker based on word probability and a grapheme-to-phoneme (G2P) algorithm are incorporated to the system output. The spell checker algorithm tries to find the most probable word sequence in a transcription within a word dictionary, thus being able to replace wrong words in a sentence by other words, while the G2P algorithm is trained on a phonetic dictionary that contains the relationship between words and their phonetic spellings The G2P algorithm is applied after the ASR transcriptions as an attempt to assess of the ASR's mistakes.

The contents of this paper are organized as follows. Sec. 1 introduces the basic end-to-end ASR model [18] based on recurrent neural networks, long short-term memories, and the CTC cost function. In Secs. 2 and 3, the spell checker and the G2P algorithms are introduced, respectively, and their implementation details are discussed. The experiments on the incorporation of the spell-checker and G2P algorithms to the ASR system are discussed in Sec. 4. Finally, conclusions are given in Sec. 5.

1 NEURAL-NETWORK-BASED ASR MODEL

Traditional neural networks do not correlate inputs with some sequential hierarchy, such as speech or any other temporal data. For such signals, one must use a recurrent neural network (RNN), which at each time step t receives at its input a new chunk of the sequence $\boldsymbol{x}^{(t)} \in \mathbb{R}^D$ and stores the current and past information in a hidden-state vector $\boldsymbol{h} \in \mathbb{R}^H$ such that

$$\boldsymbol{h}^{(t)} = \tanh\left(\boldsymbol{W}_{hx}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}\right), \quad (1)$$

where $\boldsymbol{W}_{hx} \in \mathbb{R}^{H\times D}$ and $\boldsymbol{W}_{hh} \in \mathbb{R}^{D\times D}$ are the internal network weights and $\boldsymbol{b} \in \mathbb{R}^H$ is the internal bias. Feeding the next layer with the hidden states of the current layer enables deep recurrent networks, as depicted in Fig. 1, whose Lth (last) layer's output $\boldsymbol{z} \in \mathbb{R}^C$ is such that

$$\boldsymbol{z}^{(t)} = \boldsymbol{W}_{hz}\boldsymbol{h}_L^{(t)} + \boldsymbol{b}_z, \quad (2)$$

where $\boldsymbol{W}_{h_Lz} \in \mathbb{R}^{C\times H}$ is the output weight matrix, $\boldsymbol{b}_z \in \mathbb{R}^C$ is the output bias, and $\boldsymbol{h}_L^{(t)}$ is the hidden state of layer L at time t.

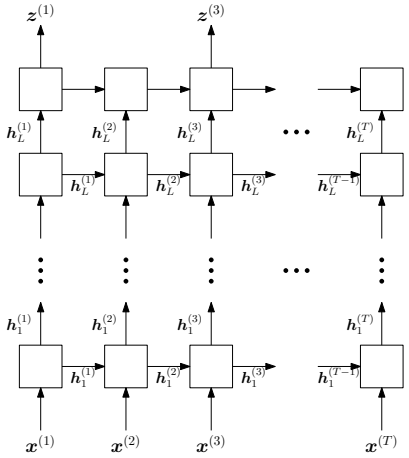

Figure 1: Example of deep RNN where output $\boldsymbol{z}$ exists only at odd timesteps.

The training of an RNN is similar to the training of traditional neural networks [1], where the outputs are evaluated against a cost function and the resulting error is backpropagated to the weights. In the RNN case, however, besides the error backpropagation through the network layers, there is also a backpropagation through time (BPTT), where the backward signal flows through the timesteps too, and a simple gradient-based optimization technique is applied.

Despite being designed to correlate samples along a sequence, the RNN training may fail in this task due to either diverging or vanishing gradient problems [21]. A harsh method to deal with gradient divergence is the

gradient-norm clipping [22], which modifies the gradient $\boldsymbol{\delta}$ according to

$$\boldsymbol{\delta} = \begin{cases} \eta \frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|_2} & \text{if } \|\boldsymbol{\delta}\|_2 \geq \eta, \\ \boldsymbol{\delta} & \text{otherwise,} \end{cases} \tag{3}$$

where η is a regularizer threshold, commonly set to $\{5,\ 10,\ 400\}$ [22, 12].

Long short-term memory [23] (LSTM) is a special kind of RNN that was explicitly designed to deal with the vanishing gradient problem. Rather than modifying the hidden state at every time step, the LSTM has four gates of size H responsible for controlling which information should persist. Moreover, it also has a cell state $\boldsymbol{c} \in \mathbb{R}^H$ that barely suffers modifications along time, attenuating the vanishing gradient problem. Mathematically, the LSTM is described by

$$\begin{gathered} \boldsymbol{a} = \boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b} = [\boldsymbol{a}_i; \boldsymbol{a}_f; \boldsymbol{a}_o; \boldsymbol{a}_g] \\ \boldsymbol{i} = \sigma(\boldsymbol{a}_i),\ \boldsymbol{f} = \sigma(\boldsymbol{a}_f),\ \boldsymbol{o} = \sigma(\boldsymbol{a}_o),\ \boldsymbol{g} = \tanh(\boldsymbol{a}_g) \\ \boldsymbol{c}^{(t)} = \boldsymbol{f} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i} \odot \boldsymbol{g},\ \boldsymbol{h}^{(t)} = \boldsymbol{o} \odot \tanh(\boldsymbol{c}^{(t)}), \end{gathered} \tag{4}$$

where $\boldsymbol{W}_h \in \mathbb{R}^{4H \times H}$ and $\boldsymbol{W}_x \in \mathbb{R}^{4H \times D}$ are the weights, $\boldsymbol{b} \in \mathbb{R}^{4H}$ is the bias, $\sigma(\cdot)$ is the sigmoid function applied element-wise, $\odot$ is the element-wise product, and $\boldsymbol{i},\ \boldsymbol{f},\ \boldsymbol{o},\ \boldsymbol{g} \in \mathbb{R}^H$ are the input gate, forget gate, output gate, and input block gate, respectively.

One of the major issues in ASR systems is the time alignment between the utterances and their corresponding transcriptions: a traditional ASR system requires a frame-level annotated transcription, which is time-consuming and computationally expensive to obtain. The so-called connectionist temporal classification [14] method was created to address this issue. The CTC consists of a softmax output layer $\hat{\boldsymbol{y}} = \exp(\boldsymbol{z}) / \sum_i \boldsymbol{z}_i$ and a different loss function. The output of the softmax layer has size $(C+1)$, where the extra class is reserved for the occurrence of the blank label $\varnothing$. Given an utterance $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(T)})$ and its label sequence $\boldsymbol{l} = (l_1, \ldots, l_u, \ldots, l_U)$, where $(2U+1) \leq T$, the CTC method tries to maximize the log-likelihood of the label sequence given the inputs

$$\mathrm{P}(\boldsymbol{p}|\boldsymbol{X}) = \prod_{t=1}^{T} \hat{y}_{p_t}^{(t)}, \tag{5}$$

where $\boldsymbol{p} = (p_1, \ldots, p_t, \ldots, p_T)$ is the CTC path, and each $\hat{y}_i, i = \{1, \ldots, C\}$ is considered independent for a given input. The CTC path $\boldsymbol{p}$ allows the occurrence of the blank label and repetitions of non-blank labels, and it is easily mapped to its corresponding label sequence $\boldsymbol{l}$ by removing the repetitions and the blank labels. Then, the likelihood of $\boldsymbol{l}$ can be evaluated as a sum of all CTC paths that can be mapped to the same label sequence $\boldsymbol{l}$, that is

$$\mathrm{P}(\boldsymbol{l}|\boldsymbol{X}) = \sum_{\boldsymbol{p} \in \mathcal{P}(l)} \mathrm{P}(\boldsymbol{p}|\boldsymbol{X}), \tag{6}$$

where $\mathcal{P}(l)$ is the set of all CTC paths associated to a given $\boldsymbol{l}$. Performing this mapping of different paths into the same label sequence is what makes it possible for CTC to use unsegmented data, as it allows the network to predict the labels without knowing when they occur.

Summing over all paths is time-consuming, which can be avoided by using the forward-backward algorithm [24]. Finally, the CTC loss function l is defined as the colog probability of correctly labeling all the training examples in some training set $\mathcal{S}$:

$$l(\mathcal{S}) = -\sum_{(\boldsymbol{X}, \boldsymbol{l}) \in \mathcal{S}} \log P(\boldsymbol{l}|\boldsymbol{X}). \tag{7}$$

This cost function is differentiable w.r.t. the outputs of the RNN $\boldsymbol{z}$ and can be used interchangeably with any other cost function for training the network parameters.

The network outputs at each timestep t a probability distribution over the set of labels $\hat{y}_i, i = \{1, \ldots, C\}$. In order to generate the label sequence $\boldsymbol{l}$, one may argue that the best way of decoding is to generate a path $\boldsymbol{p}$ by choosing the label with the highest probability at each timestep t. This greedy decoding scheme, however, does not consider that the probability of a label sequence $\boldsymbol{l}$ is the sum of all CTC paths, as shown in Eq. (6), which is the idea behind a more powerful decoding scheme called beam search decoder [25].

1.1 ASR system characteristics

The final ASR model devised in [18] has 5 bidirectional LSTM layers with 256 hidden units each. At the input, the utterances were preprocessed using 13 mel-frequency cepstral coefficients (MFCCs), calculated over a window of 25 ms with a 10-ms displacement, along with the delta and delta-delta coefficients, leading to a total of 39 features. The last recurrent layer was projected onto 28 labels ('A', 'B', ..., 'Z', 'blank label', and one special label to delimit the word boundaries), and a softmax layer was applied to score those labels.

The Brazilian Portuguese dataset [18], an ensemble of four datasets, three of them publicly available, was used to train the model. The training was carried out using the Adam [26] optimizer, with a learning rate of 10^{-3} for over 100 epochs. As a regularizer, a weight decay of 10^{-4} was applied to all weights in the model and a variational dropout [27] of 20% was applied at each layer. During the training phase, it was observed that only a few batches had a high gradient variance, which did not exceed 400; then, a gradient norm clipping with $\eta = 400$ was used just as a precaution.

The best model reported a character error rate of 24.37% on the test set, using a beam-search decoding algorithm with a beam width of 100. In the following section, all the experiments are developed over the transcription predicted by this trained model.

At each time step, the network outputs the label distribution and one may decode the sequence using either

a greedy method [28] (by choosing the label with the highest probability at time t) or by using the non-greedy beam-search decoder proposed in [25]. Even after using a more robust decoding algorithm, however, some transcriptions still contain errors, as pointed out in [20, 18]. Regarding this matter, Secs. 2 and 3 discuss two post-processing strategies devised to potentially mitigate the ASR transcriptions errors.

2 SPELL CHECKER

Given a possibly misspelled word (*i.e.* sequence of characters) w, a spell checker tries to find the correct word c^* by maximizing the probability of a candidate c being the intended correction out of all possible candidates $\mathcal{C}$, that is

$$c^* = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \, \mathrm{P}(c|w). \tag{8}$$

Using the Bayes' theorem and noticing that $\mathrm{P}(w)$ is the same for every candidate c, the probability $\mathrm{P}(c|w)$ can be determined as

$$c^* = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \, \mathrm{P}(w|c) \, \mathrm{P}(c), \tag{9}$$

where $c \in \mathcal{C}$ is the candidate model, the prior $\mathrm{P}(c)$ is the language model, and $\mathrm{P}(w|c)$ is the noisy channel model. A simple algorithm based on the mathematical model described above, inspired both on Peter Norvig's tutorial [29] and Mozilla's project Deep Speech [30], is summarized in Alg. 1.

The algorithm starts by splitting the sentence into words. Then the candidate model computes all possible words that are up to an edit (insertion, deletion, substitution, or transposition) distance of 2 from that word. The candidate model assumes that known words with lower edit distances are much more probable than words with higher edit distances. In order to make the candidate model tractable, one can restrict those word candidates to a given dictionary, and if no known candidates are found then the original word is returned.

The noisy channel model considers that every candidate at the chosen priority has the same probability. For each candidate word, new sentences are generated and then scored by the language model. The selection algorithm keeps the problem computationally tractable by removing the less probable sentences using a beam-search algorithm. Finally, the corrected sentence with the highest likelihood is returned.

The spelling correction system described here, based on statistic models, only requires a dictionary (for the candidate model) and a language model $\mathrm{P}(c)$ to work on any language. In our implementation, the Brazilian Portuguese dictionary (with almost 171,000 words) was extracted from the Google's Android repository [31]. The language model was obtained from the LAPS group [32] at the Federal University of Pará, a 3-gram model constructed with the SRILM toolkit, trained over the CETENFolha, Spoltech, OGI-22, WestPoint, LapsStory and LapsNews *corpora*, counting almost 1.6 million sentences.

Algorithm 1 Simple Spell-Checker Algorithm.

```
procedure CORRECTION(sentence)
    S ← {∅}                                   ▷ List of sentences
    P(w|c) = 1, ∀ w, c                        ▷ Noisy channel model
    for w ∈ split(sentence) do                ▷ Split into words
        for s ∈ S do
            C ← CANDIDATE MODEL(w)
            for c ∈ C do
                insert s + c into S
            end for
        end for
        score P(s) P(w|c),  s ∈ S             ▷ Using the language model and the noisy channel model
        S ← most probable sentences in S      ▷ beam search
    end for
    return argmax P(s) P(w|c),  s ∈ S
end procedure
procedure CANDIDATE MODEL(w)                  ▷ Candidates for word w
    Dictionary D
    C ← {∅}
    if w ∈ D then
        insert w into C
        return C
    end if
    C ← {edit1(w) ∈ D}                        ▷ Edit distance of 1
    if C ≠ {∅} then
        return C
    end if
    C ← {edit2(w) ∈ D}                        ▷ Edit distance of 2
    if C ≠ {∅} then
        return C
    end if
    insert w into C
    return C
end procedure
```

Consider the following sentence: "o mercado fica de alto risco a curto prazo" (the market becomes high-risk in a short time). The spell checker algorithm is able to correct mistakes in different words up to 2 errors as in "o mercado fica de alt**u** ri**zc**u a curto prazo" where the word "alto" has one substitution and the word "risco" has 2 substitutions. However, if the "alto" is changed to "autu", the algorithm will prioritize known words of smaller edit distance from the wrong word, misleading the word "autu" to "auto' (self), for instance. Nevertheless, in this last sample, the words "alto" and "autu" are near-homophones, *i.e.*, they may lead to the same phonetic transcription.

3 GRAPHEME-TO-PHONEME CONVERTER

The grapheme-to-phoneme (G2P) conversion is a process that generates a pronunciation for a word, and has several applications in text-to-speech and speech recognition. To the authors' best knowledge, the top performing methods for G2P conversion are based on joint-sequence models [33] and sequence-to-sequence (seq2seq) models [17].

Joint-sequence models divide the G2P problem into three parts: alignment (between graphemes and phonemes); training (learning the grapheme-phoneme conversions); and decoding (finding the most probable pronunciation given the model). This model type tries to estimate grapheme-phoneme alignments, which in practice is not always straightforward [34].

Seq2seq G2P models [35] are based on recurrent neural networks (RNN) that transform an input sequence (*e.g.* word) into another sequence at the output (*e.g.* phonetic transcription), without any input-output alignment requirement. This transformation is accomplished by using an encoder — an RNN that maps the input sequence into a vector — and a decoder — another RNN that maps the encoded vector into the desired phonetic translation — as illustrated in Fig. 2. Seq2seq models are easier to implement and have the advantage of using a continuous space representation of words, where words in a similar context tend to appear close to each other in the representational space, thus enabling neural-based models to better generalize to unseen words.

Both G2P models described here were custom developed. The Sequitur tool [36], distributed under the GNU Public License, was selected for the joint-sequence model, whereas the CMU Sphinx G2P toolkit [37], distributed under Apache license, was employed for the seq2seq model. The G2P models were trained using the LAPS phonetic dictionary containing over 64,000 words, comprising 38 phones based on the SAMPA alphabet, separating 10% of this training set for test and 5% for validation. The Sequitur model was trained using the default parameters, and the CMU Sphinx G2P was trained using 2 LSTM layers with 512 hidden units each.

The Sequitur tool and the CMU Sphinx G2P model reported a WER of 1.79% and 1.40% on their respective test sets. Most incorrect conversions were quite plausible and were due to proper or foreign words, like "jonas" (converted to /Z o n a s/ instead of /Z o$\sim$ n a s/) and "named" (converted to /n a m e dZ/ instead of /n a$\sim$ m e dZ/). Hence, both models yielded very similar results and presented a very low WER. From now on, only the results for the CMU Sphinx G2P model are reported, which achieved the lowest WER.

4 ASR EXPERIMENTS WITH POST-PROCESSING MODULES

4.1 Spell-checker experiment

In this experiment, we investigate the effect of incorporating a spell checker to the deep-learning ASR system developed in [18] for Brazilian Portuguese. When using the standard spell checker described in Section 2, the CER increased from 24.37% to 24.71%, indicating that the ASR original output included word errors beyond an edit distance of 2. Unfortunately, it is not possible to extend the spell-checker algorithm to comprise distances above 2 as the computational cost exponentially increases with this threshold.

A possible issue associated to the spell-checker Alg. 1 is the absolute priority of the candidate model with lower edit distances, which forces the algorithm to choose corrections with distance 1, even if there is a better candidate (with higher probability, as given by the language model) with edit distance 2. Therefore, an algorithm modification was proposed to give a chance for candidates with higher edit distances: instead of considering $P(w|c) = 1\ \forall w, c$, the new candidate model re-scores $P(w|c)$ by giving different weights w_d for candidates with distinct edit distances d. The CER results for the ASR system incorporating such modification are depicted in Fig. 3 for $w_1 = 1$ and different values of $0 \leq w_2 \leq 1$. Unfortunately, giving a chance to higher edit distances brings no improvement to the ASR CER performance.

After analyzing some spell-checker outputs, as shown in Tab. 1, one concludes that the main issue with this algorithm is that it operates on a word basis and not at the sentence level. Therefore, the algorithm tries to correct a word instead of searching for the most probable sentence. In Tab. 1, the network failed to insert the correct word boundaries in "ideia da qualidade", and the spelling correction found that "idede" is more probable to be "idade" (1 edit distance) than "ideia de", which would be more compatible with the whole sentence. When there are only simple missing/replacements as in "importanes" and "cualidade", the algorithm is almost always able to correct the words. However, on the average, the spell checker was not able to reduce the CER mostly due to the wrong word boundaries inserted by the network. This issue can be mitigated by a smarter decoder that takes into account the language model, or by training over a domain distinct from the characters, as considered in the next experiment.

4.2 Grapheme to phoneme experiment

In this experiment, we investigate the ASR performance on the phoneme level instead of the character level assessed by the CER. As said above, we use a 38-phoneme representation for Brazilian Portuguese, which leads to 39 phoneme-level labels, including the space, as opposed to the 28-label character space. Then,

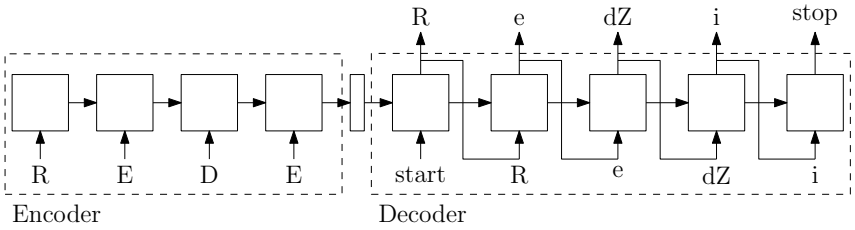

Figure 2: A simple G2P seq2seq model. The encoder transform the word input 'REDE' into a vector representation; the decoder transforms this vector into the phonetic representation '/R e dZ i/'.

Table 1: Comparison between transcriptions predicted by the network solely and after using a spelling correction, and the ground truth.

	Transcriptions
Ground Truth	e muito **importante divulgar** a **ideia da qualidade** para o publico
Predicted	e muito **importanes de vulga idede cualidade** para o podo
Spelling correction	e muito **importantes de julga idade qualidade** para o podo

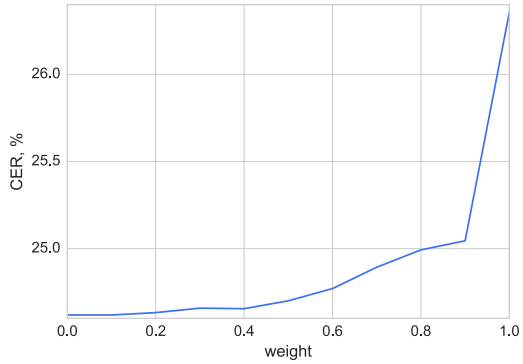

Figure 3: CER for different values of $0 \leq w_2 \leq 1$ with $w_1 = 1$.

each utterance is translated to a set of phonemes using the G2P converter after the network's transcriptions, and all performance evaluation regarding the G2P is done on the phoneme space.

We then consider the standard ASR system followed by a G2P converter or by a spell-checker/G2P combination. Results for the normalized label error rate (NLER), which takes into consideration the number of possible labels either in the phoneme or the character space, are summarized in Tab. 2, which shows that many of the ASR mistakes are phonetically reasonable, and can be alleviated by a G2P system. In fact, using the G2P system, one gets a phoneme error rate of 27.26% in a set of 39 labels, which corresponds to a NLER in the 28-label space of $27.26 \times 28/39 = 19.57\%$.

When one uses a spell-checker/G2P combination, the NLER result increases slightly to 20.08%. This can be explained by the spell-checker algorithm changing the word pronunciation significantly, since this aspect is not taken into account for the word selection in its implementation, as indicated in the last line of the Tab. 1.

Table 2: Label error rate (LER) for different post-processing schemes. The normalized LER indicates the error rate in the 28-label space.

Algorithm	Edit distance	Normalized LER	# labels
Baseline	-	24.37%	28
Spell checker	2	24.71%	28
G2P	-	**19.57%**	39
Spell Checker + G2P	2	20.08%	39

5 CONCLUSIONS

This paper investigated the results of two simple post-processing techniques applied to a deep-learning based ASR developed for Brazilian Portuguese. It was verified that a simple spell-checker algorithm was not capable of correcting the model mistakes due the wrong word boundaries inserted by the ASR system. The errors, however, are phonetically reasonable and can be alleviated by using a label set in the phonetic domain. Nevertheless, concatenating the two post-processing schemes brings no improvements, since the spell-checker algorithm does not account for correct words with similar sounds, jeopardizing the G2P algorithm performance. One then concludes that using a language model to better decode the outputs of an end-end ASR system is indispensable: lower word-boundary errors enable the spell-checker algorithm to perform word-level fine corrections. It was also observed that training over the phonetic space, instead of the character level, also increases the accuracy of the system, due to the non-overlapping label set domain.

ACKNOWLEDGMENT

The authors would like to thank CAPES, CNPq, and FAPERJ Brazilian agencies for funding this work.

REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, Adaptive Computation and Machine Learning. MIT Press, Cambridge, England, 2016.

[2] Wanli Ouyang and Xiaogang Wang, "Joint deep learning for pedestrian detection," in *IEEE International Conference on Computer Vision*, Sydney, Australia, December 2013, pp. 2056–2063.

[3] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, USA, June 2014, pp. 1701–1708.

[4] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, January 2017.

[5] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," October 2016, eprint arXiv:1609.08144v2.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, Lake Tahoe, USA, December 2012, pp. 1097–1105.

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems*, Long Beach, USA, December 2016, pp. 379–387.

[8] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei, "Fully convolutional instance-aware semantic segmentation," November 2016, eprint arXiv:1611.07709v1.

[9] Geoffrey E Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, November 2012.

[10] G Zweig, J Droppo, W Xiong, A Stolcke, X Huang, F Seide, M Seltzer, and D Yu, "The Microsoft 2016 conversational speech recognition system," January 2017, eprint arXiv:1609.03528v2.

[11] George Saon, Tom Sercu, Steven Rennie, and Hong-Kwang J Kuo, "The IBM 2016 English conversational telephone speech recognition system," April 2016, eprint arXiv:1604.08242v2.

[12] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Awni Y Hannun, Billy Jun, Tony Han, Patrick LeGresley, Xiangang Li, Libby Lin, Sharan Narang, Andrew Y Ng, Sherjil Ozair, Ryan Prenger, Sheng Qian, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Chong Wang, Yi Wang, Zhiqian Wang, Bo Xiao, Yan Xie, Dani Yogatama, Jun Zhan, and Zhenyao Zhu, "Deep speech 2: end-to-end speech recognition in English and Mandarin," in *International Conference on Machine Learning*, New York, USA, June 2016, vol. 48, pp. 1–10.

[13] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig, "Achieving human parity in conversational speech recognition," February 2016, eprint arXiv:1610.05256v2.

[14] Alex Graves, Santiago Fernández, Faustino J Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *International Conference on Machine Learning*, Pittsburgh, USA, June 2006, pp. 369–376.

[15] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems*, Montreal, Canada, December 2015, pp. 577–585.

[16] "Google voice search: Faster and more accurate," https://goo.gl/rvYqLH, Accessed: 2017-03-02.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Pro-*

cessing Systems, Montreal, Canada, December 2014, pp. 3104–3112.

[18] Igor Macedo Quintanilha, "End-to-end speech recognition applied to Brazilian Portuguese using deep learning," M.Sc. dissertation, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2017.

[19] Igor Macedo Quintanilha, Luiz Wagner Pereira Biscainho, and Sergio Lima Netto, "Towards an end-to-end speech recognizer for Portuguese using deep neural networks," in *XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, São Pedro, Brazil, September 2017, Accepted.

[20] Andrew L Maas, Quoc V Le, Tyler M O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Annual Conference of the International Speech Communication Association*, Portland, USA, September 2012, pp. 22–25.

[21] Yoshua Bengio, Patrice Y Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, March 1994.

[22] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, Atlanta, USA, June 2013, vol. 28, pp. 1310–1318.

[23] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[24] Lawrence R Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[25] Andrew L Maas, Ziang Xie, Dan Jurafsky, and Andrew Y Ng, "Lexicon-free conversational speech recognition with neural networks," in *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, USA, May 2015, pp. 345–354.

[26] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations*, San Diego, USA, May 2015, pp. 1–15.

[27] Yarin Gal and Zoubin Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems*, Long Beach, USA, December 2016, pp. 1019–1027.

[28] Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Studies in Computational Intelligence. Springer Verlag, Heidelberg, Germany, 2012.

[29] Peter Norvig, "How to write a spelling corrector," http://norvig.com/spell-correct.html, Accessed: 2017-05-29.

[30] Mozilla, "A Tensorflow implementation of Baidu's deepspeech architecture," https://github.com/mozilla/DeepSpeech, Accessed: 2017-05-29.

[31] Google, "Google's Android repository," https://android.googlesource.com, Accessed: 2017-05-29.

[32] "Falabrasil - UFPA," http://www.laps.ufpa.br/falabrasil/, Accessed: 2017-03-06.

[33] Maximilian Bisani and Hermann Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, May 2008.

[34] Kanishka Rao, Fuchun Peng, Hasim Sak, and Françoise Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brisbane, Australia, April 2015, pp. 4225–4229.

[35] Kaisheng Yao and Geoffrey Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," in *Annual Conference of the International Speech Communication Association*, Dresden, Germany, September 2015, pp. 3330–3334.

[36] RWTH Aachen university, "Sequitur G2P," https://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html, Accessed: 2017-05-31.

[37] CMUSphinx, "CMUSphinx," https://cmusphinx.github.io, Accessed: 2017-05-31.