

Using Error-Driven Approach to Improve Automatic Grapheme-to-Phoneme Conversion Accuracy

Tatyana Polyakova, Antonio Bonafonte

Technical University of Catalonia
Jordi Girona 1-3, Barcelona, Spain
tatyana@gps.tsc.upc.es, antonio.bonafonte@upc.edu

Abstract

In speech technology it is very important to have a system capable of accurately performing grapheme-to-phoneme conversion, which is not an easy task especially if talking about languages like English where there is no obvious letter-phone correspondence. Manual rules so widely used before are now leaving the way open for the machine learning techniques.

In this paper we present an extension of the use of transformation-based error-driven algorithm to grapheme-to-phoneme task. A set of explicit rules was inferred to correct the pronunciation predicted by three well-known machine-learning techniques and by simply assigning to the letter the most likely phone that has been seen for this particular letter in the training corpus. All the methods applied in combination with fnTBL toolkit (Florian and Ngai, 2001) significantly outperform the results obtained by these methods alone.

To evaluate the transcription two types of error rate are calculated: the word error rate and the phoneme error rate. The word error rate is the most important as it shows the real potential of a transcription system to grasp the language behavior.

1. Introduction

For languages with deep orthography the problem of grapheme-to-phoneme conversion still remains unsolved and deserves to be treated.

The language growth is a non-stop process; everyday new words are entering from other languages making it impossible to have a manually transcribed lexicon including all the words.

The grapheme-to-phoneme (g2p) conversion forms a very important part of the text-processing module of text-to-speech synthesis systems, where any errors are highly undesirable because a bad start decreases the overall performance of any system. In order to get good quality speech at the output, one cannot begin to synthesize with a high percentage of erroneous transcriptions. In TTS the presence of the errors in grapheme-to-phoneme conversion module is much more critical than in ASR.

Facing the age when the methods of automatic data acquisition are the most economic in terms of time and effort, the hand-written corpus labeling is being substituted by the automatic ML methods.

A review of automatic grapheme-to-phoneme techniques can be found in (Damper et al., 1999).

The ML methods applied up to now to the problem leave a way for some further improvements. In English errors mostly appear when assigning pronunciation to vowels, whereas consonants are usually predicted better.

To improve the performance of the grapheme-to-phoneme transcription system, in this paper we propose to use an approach based on learning from errors committed by another conversion system used previously.

The transformation-based error-driven learning algorithm invented by Brill (1995) was successfully applied to such NLP tasks as part-of-speech tagging, word sense disambiguation, phrase chunking etc. The high level of accuracy achieved for these tasks proves the effectiveness of this data-driven method.

To obtain the necessary initial prediction and to train the error-driven system we use four baseline conversion methods in our experiments: CART (Black et al., 1998), FST, (Galescu and Allen, 2001), HMM with a configuration proposed by Taylor (2005) and such a naive prediction as the most-likely phone.

2. Conversion Approaches

2.1. Alignment

Many of the used methods require letters and phonemes to be aligned in a way that each letter corresponds to exactly one phoneme. Then any classifier can be applied to map graphemes to phonemes. For the words which have fewer letter than phonemes an empty phoneme “ε” was introduced.

The manual alignment, the one where the letter-phoneme correspondence is the most logical is very expensive and non-transferable to other languages. For this reason, in this paper the orthographic word representation was aligned automatically with the corresponding phonetic one using the epsilon-scattering alignment method (Black et al., 1998). In the first place all the alignments candidates were considered and then by applying the expectation maximization algorithm (Dempster et al., 1977) the best candidate was chosen.

2.2. Decision Trees (DT)

A classical machine learning technique often applied to obtain grapheme-to-phoneme transcriptions is the method based on decision trees. It was proposed by Black et al., (1998) with its posterior incorporation into Festival speech synthesizer.

A decision tree has as the input grapheme sliding window with three letters to the left and three to the right accordingly. This method is appropriate for discreet characteristics and produces rather compact models, whose size is defined by the total number of questions and leaf nodes in the output tree.

Usage only of grapheme context both on left and the right side by DT has a disadvantage: it assumes that the

decisions are independent one from another so is that it cannot use the prediction of the previous phone as the reference to predict the next one. Another limitation introduced by the binary decision trees that every time a question is asked the training corpus is divided into two parts and further questions are asked only over the remaining parts of the corpus.

2.3. Finite State Transducers (FST)

This approach chooses the pronunciation φ that maximizes the probability of a phoneme sequence given the letter sequence g .

$$\hat{\varphi} = \arg \max_{\varphi} \{p(\varphi/g)\} \quad (1)$$

In this paper a finite state transducer similar to that of Galescu and Allen, (2001) has been used.

To estimate the probability (1) is the same as to estimate the probability of the grapheme-phoneme pair, given a letter sequence. This estimation can be done using standard n-gram methods. Grapheme-phoneme pairs are extracted from the aligned dictionary.

$$p(g, \varphi) = \prod_{i=1}^N p(g_i, \varphi_i / g_1^{i-1}, \varphi_1^{i-1}) \quad (2)$$

where N is the number of letters in the word.

N-grams can be represented by a finite-state automaton, where a new state is defined for each history h and a arc is created for each new grapheme-phoneme pair (g, φ) . These arcs are labeled with a grapheme-phoneme pair and weighted with the probability of (g, φ) given the history h . To derive the finite state transducer the labels attached to the automaton edges are split in a way that letters become input and phonemes become output.

Solving the equation (2) is equivalent to finding the best path through the FST maximizing the probabilities, given g ; this is done by means of dynamic programming.

To allow maximum flexibility, the x-gram was used (Bonafonte, Mariño, 1996). The x-gram is an n-gram with flexible length. In this model the length of conditioning history depends on each particular history. Choosing x-gram's parameters carefully the number of states can be significantly reduced without any decrease in model's performance.

2.4. Hidden Markov Models (HMM)

It was recently proposed by Taylor (2005) to use hidden Markov models to confront the difficult problem of phoneme prediction.

In this case each phoneme is represented by one HMM and letters are the emitted observations.

The probability of transitions between models is equal to the probability of the phoneme given the previous phoneme. The objective of this method is to find the most probable sequence of hidden models (phonemes) given the observations (letters), using the probability distributions found during the model training.

$$\hat{\varphi} = \arg \max_{\varphi} p(g, \varphi) p(\varphi) \quad (3)$$

where $p(\varphi)$ is the probability of phoneme sequence, and $p(g, \varphi)$, is the grapheme-phoneme joint sequence probability.

One model is trained for each phoneme; the maximum number of letters that a phoneme is able to generate was taken to be four, since it is uncommon that more than four letters represent a single sound, at least in English. No looping states are allowed unlike in the model configuration that serves for speech recognition.

For this method the grapheme-to-phoneme alignment of the dictionary is not necessary as it is done during the training stage by Baum-Welch training (Jelinek, 1998) in which the HMM uses the probabilities of the grapheme-to-phoneme correspondence found in the previous step of the algorithm.

The automatic speech recognition toolkit was used to train the HMM models and to decode graphemes into phonemes; the models were enhanced by phoneme x-gram of maximum length of 5 to benefit from the information about the phoneme context which was proved to be necessary.

3. Advantages of Transformation-Based Error-Driven Approach Learning (TBL)

The transformation-based error-driven algorithm (TBL) originally invented by Brill (1995) consists in learning the transformation rules from the training data that is labeled with some initial classes.

The main difference between manually derived set of rules and the set of rules extracted by TBL is that the second set doesn't need to be elaborated by experts. The method is fully automatic apart from the rule template creation step.

The order of rule application also does not require any knowledge about the language. It is established automatically during the training of the system. The rules that have the highest best score are put at the top of the list and then the other ones with a lower score are added.

The rules are language independent and could be applied to any supervised prediction task in combination with any machine learning technique, while the manually elaborated rules are non-transferable to other languages, which is an added disadvantage to their very high development cost.

As CART or FST, this method requires the data to be aligned in a one-to-one manner. Since sometimes the data alignment is not unique it introduces a limitation in respect to as to what errors could be corrected by rules.

During the training process the algorithm's main goal is to capture certain regularity between the errors in the first prediction and to choose best transformation rule according to the environment where the error was committed.

The learning process is similar to when a human is trying to learn a language, a human learns from errors by memorizing certain conditions under which the error was committed, in the future trying to avoid the occurrence of the same error, given alike circumstances. If compared to foreign language learning many examples of similar situations can be found which could be incorrect word order in the question, erroneously memorized noun gender (for languages where it is present), stress misplacement, etc.

This learning mechanism is activated every time the error is committed. It could be applied not only to

language learning but also to many areas of human activities. It is very easy to make a mistake, but after knowing the right way to do it, it is unlikely to repeat the same or similar mistake given the same conditions. The TBL algorithm works in the same way: it generates rules that try in the best way to generalize the transcription errors obtained by the initial prediction method. Once the patterns *transformation_condition*→*correct_answer* are captured, the TBL applies these patterns to correct the errors. The error itself usually forms part of the transformation condition as well as the conditions of its occurrence.

In this paper we present the comparative analysis of applying transformation-based learning algorithm to correct the prediction inferred by other machine learning techniques.

4. Applying TBL to G2P conversion task.

Applying the transformation-based learning to grapheme-to-phoneme task seems to be of great advantage because it is completely language independent, efficient and easy to understand.

Using the transformation-based error-driven learning algorithm to correct the prediction previously obtained by another classifier permits us to capture the imperfections of the previous approximation to the linguistic irregularities into a set of context-dependent transformation rules, where the context serves as the conditioning features.

If at the input set of features coincides with the features defined in the rule, then the rule is applied to the entire set of features and then its score is calculated depending on the number of the errors the rule has been able to correct. The best rule is therefore chosen.

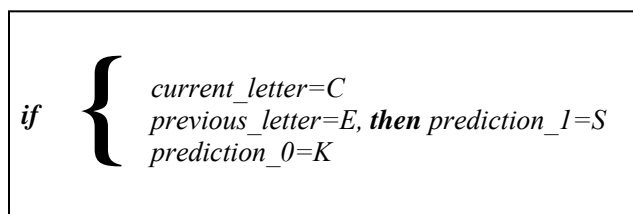


Figure 1: Example of a rule. The transformation to be performed under these conditions is to assign the *prediction_1* a new value, which in this case will be *s*.

The rules are generated at each step, first each one of them is applied to the corpus initialized with some prediction, then the best rule found is applied to the training corpus and the second prediction is obtained and so on. In this optimization task the objective function is error rate. The process continues until no rule that improves the accuracy could be found or a rule with a score lower than the preset threshold has been generated. Another advantage of the transformation-based learning algorithm is that it uses previously corrected predictions to generate new prediction transformation, making these more reliable as to say it uses intermediate results to find a better way to correct the prediction.

Rules are the main factor influencing on the overall performance. Rule templates define what set of rules is to be generated and applied to the corpus in search of the one that best corrects the misprediction. An example of rule templates is given in Figure 2.

```

let_-1 let_0 let_1 => fon
let_-1 let_0 let_-1 fon_0 => fon
let_-2 let_-1 let_0 let_1 fon_0 => fon
let_-3 let_-2 let_-1 let_0 let_1 fon_0 => fon
let_-3 let_-2 let_-1 let_0 let_1 let_2 let_3 fon_0 => fon
let_-1 let_0 let_1 fon_-1 fon_0 fon_1 => fon

```

Figure 2: Example of a rule template.

To create the rule templates for the experiment we employed most of all the possible combinations of the letter and phone contexts, limiting the largest letter context to be plus/minus 5 letters and largest phone context to be plus/minus 3 phones, correspondingly.

To obtain the results the fnTBL toolkit, kindly provided for public use by its authors (Florian and Ngai, 2001) was used. The fnTBL differs from the original Brill's TBL in the way that the objective function is calculated and reaches a speed up, without reducing the system's performance. The Figure 3 shows a scheme of algorithm combination.

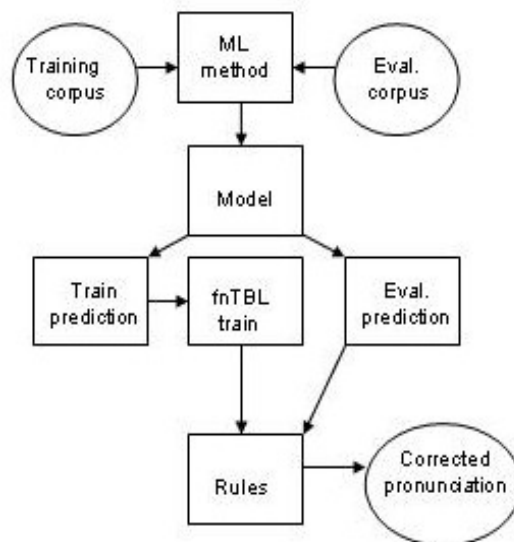


Figure 3: Scheme of combination of ML learning techniques with fnTBL algorithm.

5. Experimental results

In the experimental section we present the outcome of combined application of various classifiers to the grapheme-to-phoneme task as well as the results obtained by each classifier alone. The experiments were conducted using two lexicons : the LC-STAR dictionary of American

English covering about 50,000 words, produced by NSC (Natural Speech Communications) in the framework the LC-STAR project (www.lc-star.com); and the Unisyn lexicon that comprises all the rich variety of different pronunciations of English including standard American English which was taken from the publicly available source (www.cstr.ed.ac.uk); it has about 110,000 words. The second dictionary consists of English words of ordinary usage while the LC-STAR dictionary includes more specific terms and some foreign words. The training and test data set are 90% and 10% percent accordingly.

5.1. G2P results

In the Table 1 the results obtained for the LC-STAR dictionary by three statistical classifiers are given.

	Phon.	Words
DT	93.93%	68.32%
FST	93.63%	75.66%
HMM	84.16%	47.54%

Table 1: Percentage of correct phonemes and words for the LC-star dictionary.

The Table 2 shows the corresponding results for the Unisyn lexicon.

	Words	Phon.
DT	72.67%	95.26%
FST	86.65%	97.30%
HMM	54.87%	86.93%

Table 2: Percentage of correct phonemes and words for the Unisyn lexicon.

Finite-state transducers give much higher word accuracy than hidden Markov models or decision tree for both lexicons. The method giving the poorest results is the one based on HMM. To improve these results some kind of preprocessing might be needed like that proposed by Taylor (2005).

Applying a number of rewrite rules to the dictionary, using context-sensitive models together with introducing of stress patterns, in the case of a stress lexicon, allow obtaining better results with HMM.

After obtaining the results above the fnTBL algorithm was combined, with all the methods used above to correct the pronunciation predicted by them before following the scheme shown in the Figure 3. Also, such a naïve prediction as assigning the most-likely phone seen in the training to each grapheme accordingly was considered.

Tables 3 through 6 show the percentage of the correct phonemes and words as a result of combination of four classifiers with learning from errors algorithm.

Its goal was to learn rules that would be able to correct the prediction obtained previously. The rules were learned for 3 different sizes of letter context: the maximum letter context included in the rules varied from 3 to 5 letters to the left and/or right.

	baseline	cont=3	cont=4	cont=5
ML	59.70%	95.17%	95.59%	95.76%
DT	93.93%	94.64%	94.85%	95.00%
FST	93.63%	95.73%	95.87%	95.92%
HMM	84.16%	92.66%	93.15%	93.29%

Table 3: Percentage of correct phonemes for the combination of 4 methods with fnTBL for LC-STAR lexicon.

	baseline	cont=3	cont=4	cont=5
ML	1.07%	75.67%	77.46%	78.26%
DT	68.32%	73.06%	74.13%	74.68%
FST	75.66%	78.79%	79.33%	79.63%
HMM	47.54%	67.01%	68.70%	69.08%

Table 4: Percentage of correct words for all the prediction methods enhanced by fnTBL, the results are given for the LC-STAR lexicon.

	baseline	cont=3	cont=4	cont=5
ML	56.36%	96.68%	97.01%	97.12%
DT	95.26%	96.44%	96.60%	96.67%
FST	97.30%	97.46%	97.47%	97.49%
HMM	86.93%	94.38%	94.45%	94.75%

Table 5: Percentage of successfully predicted phones by combining all four prediction methods with fnTBL for Unisyn lexicon.

	baseline	cont=3	cont=4	cont=5
ML	1.42%	82.39%	84.00%	84.61%
DT	72.67%	80.74%	81.63%	82.08%
FST	86.65%	87.25%	87.28%	87.36%
HMM	54.87%	74.19%	74.32%	74.95%

Table 6: Percentage of successfully predicted words by combining all four prediction methods with transformation-based learning algorithm for Unisyn lexicon.

Applying the error-correcting rules to the output of various algorithms shows us a significant improvement of those results.

The biggest improvement was achieved for the methods whose performance at the start was the poorest; it is due to the fact that the abundance of errors gave a way to their better generalization and encapsulation into the transformation rules. The decision tree results were improved by a measure of 8-10 %, the HMM results were improved by 20 %, and the FST's prediction improvement range is 1-5 %. The hugest improvement was made for the most likely phone prediction, where the preliminary prediction scored about 50 % phonemes and 1% words correct. Among the correctly predicted phonemes there were mostly consonants. The improvement ranged from 75 to 80 % and the results are similar to those obtained by combining fnTBL with the best performing method, FST, which shows the effectiveness of the trained rules. The improvement was the less context-sensitive the better was the baseline prediction. The largest context gave the best results although it was more expensive in terms of the computation time. The time needed for computation also depended on the number of baseline errors. If there were

few errors to correct in the training corpus the algorithm converged faster.

Figure 4 shows the distribution of the number of errors per word before and after the application of the TBL algorithm to correct pronunciation. This analysis was done for the HMM method and the LC-STAR lexicon, which seem to give more problems.

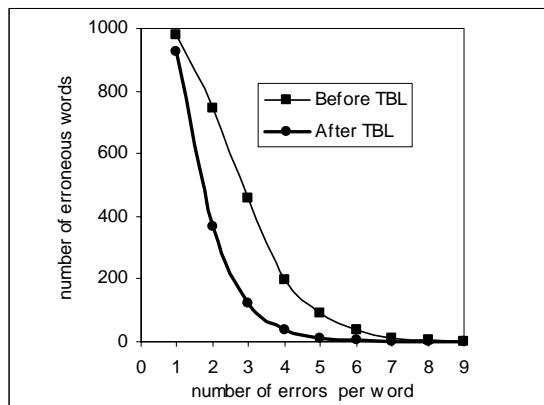


Figure 4: Number of words as a function of number of errors per word.

From the Figure 4 it can be seen that the application of fnTBL reduces the number of words with severe pronunciation errors (more than 3 errors per word) by almost seven times and the rest of errors by 1,5 times. Words with only one error are still numerous and it could be due to the fact that these errors are very specific, usually these errors are a consequence of the confusion between different vowels represented by the same letter. These are the language irregularities that are very difficult to capture as well for the automatic algorithms that try to pronounce English text as for English language learners (humans). The generalization capacity of the fnTBL is also limited by the inconsistency of some alignments and of the reference transcriptions. It should be taken into account that the error rate ranging from 11%-21% in the g2p experiments was obtained for only out-of-vocabulary words. Therefore, the percentage of pronunciation errors in the complete system is definitively smaller.

6. Conclusions

The goal to improve grapheme-to-phoneme conversion results was set and achieved by means of applying a set of transformations learned from errors. The transformation rules were learned automatically from a training corpus previously labeled using four classifiers. The rule templates are language independent and can be used to generate transformation rules for any language.

The combination of all methods with transformation-based error-driven algorithms significantly improved the results obtained by these methods alone.

Correcting the errors in the case where the most-likely phone was assigned to the letter also gave competitive results proving the effectiveness of the transformations rules. In fact the results were higher than those obtained by the widely used decision trees and by newly proposed HMM for both dictionaries.

In the future the algorithm will be applied to predict the stress and the influence of such information as part-of-speech tags on the conversion results will be studied.

7. Acknowledgements

This work was sponsored by the European Community in the frameworks of TC-STAR project (IST-2002-FP6-506738, <http://www.tc-star.org>). The authors would also like to thank Radu Florian for helpful comments.

8. References

- Black A., K. Lenzo K. and V. Pagel. (1998). Issues in building general letter to sound rules. *In Proc. of the 3rd ESCA workshop on speech synthesis.*, Jenolah Caves, Australia, pp. 77-80
- Bonafonte A. and J. B. Mariño. (1996) Language modeling using X-grams. *In Proc. of ICSLP-96*, Vol.1, Philadelphia, pp. 394-397
- Brill E., (1995) Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational linguistics 21(4)*, pp. 543-565.
- Damper R. I., Marchand Y., Adamson M. J. and Gustafson K. (1998). A comparison of letter-to-sound conversion techniques for English text-to-speech synthesis. *In Proc. of the Institute of Acoustics 20(6)*.
- Dempster A. P., Laird N. M. and Rubin D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Ser. B*, 39, 1-38.
- Galescu L., J. Allen. (2001) Bi-directional Conversion Between Graphemes and Phonemes Using a Joint N-gram Model, *In Proc. of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Perthshire, Scotland.
- Florian R., Ngai G. (2001) Transformation-based learning toolkit, John Hopkins University.
- Jelinek F. (1998) Statistical Methods for Speech Recognition, *MIT Press*.
- Ngai G., Florian R., (2001) Transformation-based learning in the fast lane. *In Proc. of North American ACL*, pp. 40-47.
- Taylor P. (2005). Hidden Markov Models for grapheme to phoneme conversion, *In Proc. of Interspeech 2005*, Lisbon, Portugal, pp. 1973-1976