Comparison of Grapheme-to-Phoneme Methods on Large Pronunciation Dictionaries and LVCSR Tasks

Stefan Hahn[1]*, Paul Vozila*[2]*, Maximilian Bisani*[2]

[1]Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, 52056 Aachen, Germany
[2]Nuance Communications, Inc., 1 Wayside Road, Burlington, MA 01803, United States
hahn@cs.rwth-aachen.de, {paul.vozila,maximilian.bisani}@nuance.com

Abstract

Grapheme-to-Phoneme conversion (G2P) is usually used within every state-of-the-art ASR system to generalize beyond a fixed set of words. Although the performance is typically already quite good ($< 10\%$ phoneme error rate) and pronunciations of important words are checked by a linguist, further improvements are still desirable, especially for end user customization.

In this work, we present and compare five methods/tools to tackle the G2P task. Although most of the methods have already been published and/or are available as open source software, the reported experiments are done on large state-of-the-art tasks and the used software is from the actual publications.

Besides an experimental comparison on text data for a range of languages (i.e. measuring the G2P accuracy only), our focus in this paper is measuring the effect of improved G2P modeling on LVCSR performance for a challenging ASR task. Additionally, the effect of using n-Best pronunciation variants instead of single best is investigated briefly.

Index Terms: grapheme-to-phoneme conversion, G2P, ASR

1. Introduction

Over the years, many methods have been published to tackle the grapheme-to-phoneme conversion (G2P) task. This task is usually defined as follows: Given an orthographic form of a word (grapheme sequence $\boldsymbol{g}$), the corresponding most likely pronunciation (phoneme sequence $\boldsymbol{\varphi}$) is:

$$\boldsymbol{\varphi}(\boldsymbol{g}) = \underset{\boldsymbol{\varphi}' \in \Phi^*}{\operatorname{argmax}}\, p(\boldsymbol{g}, \boldsymbol{\varphi}')$$

Here, a grapheme $g \in \boldsymbol{g}$ is defined as a symbol used for writing language (e.g. a letter) and a phoneme $\varphi \in \boldsymbol{\varphi}$ as the smallest contrastive unit in the sound system of a language.

G2P is a task from the group of monotone string-to-string translation problems, which also includes part-of-speech tagging, name transliteration [1], and concept tagging (NLU)[2]. Thus, all the described methods could also be used to tackle other tasks from this group.

Most of the published, statistical approaches to the G2P task can be decomposed into three sub-problems. As training material, usually a corpus is given containing corresponding pairs of orthographies and phoneme sequences. In a first step, an alignment is generated between graphemes and phonemes, since it is usually not provided within the training data, e.g.:

"phoenix"	=	ph	oe	n	i	x
finɪks		f	i	n	ɪ	ks

The resulting blocks of aligned tokens are typically referred to as joint-multigrams, grapheme-phonemes, graphonemes, or graphones for short in the literature and have been introduced in [3]. The length for the number of graphemes/phonemes per graphone may be restricted and empty tokens may be allowed on either side, depending on the alignment algorithm.

This alignment may be calculated prior to the training of the statistical model and kept fix or a re-alignment step may be included within the training process which would be the next step. Here, mostly methods based on n-grams are used. The final step is the decoding, which determines how a given model is used to generate a phoneme sequence given a grapheme sequence.

One of the requirements of a G2P system for this work was that training and decoding can be done in reasonable time on large data sets. Thus, we did not use computational expensive methods like discriminative methods based on e.g. Conditional Random Fields (CRFs) [2] or online discriminative training as presented in [4].

The remainder of the paper is structured as follows: in the next section, we will present the theoretical background to the five methods which we did compare. The following section will present experimental findings, both, on text data only as well as integrated into a contemporary LVCSR system. The paper concludes with a summary of our findings.

2. Methods

In this section, we present the five methods used for the experimental comparison. The technical background is presented briefly with pointers to reference publications except for the first method which has been only documented in an in-house technical report. We applied the actual software used in the referenced publications, which in some cases is also available to the public (open source).

2.1. Combined n-Gram and Decision Tree Model (ngdt)

Within this in-house method proposed in [5], the alignment between graphemes and phonemes is generated using a variant of the Baum-Welch expectation maximization (EM) algorithm. The initialization resembles the first Baum-Welsh iteration with uniform initial distributions. To obtain these distributions, all possible alignments not mixing deletions and insertions are averaged. Another constraint is that, for simplicity, only $1 : N$, e.g. grapheme-*unit* phoneme-*sequence* alignments are allowed. After convergence, the alignment and thus the resulting graphones are kept fixed for the actual model training.

To build the n-gram model, maximum likelihood (ML) es-

timators are used on graphone sequences $\boldsymbol{q}$:

$$Pr(\boldsymbol{g},\boldsymbol{\varphi}) = Pr(\boldsymbol{q}) = \prod_{i=1}^{N} Pr(q_i|q_{i-1},\ldots,q_1) \qquad (1)$$
$$= P_{ng(i)}(q_i|q_{i-n+1},\ldots,q_{i-1})$$

To incorporate lower-order n-grams, their ML estimators are linearly interpolated using normalized interpolation scales α_i:

$$P_{ng}(\cdot) = \prod_{i=0}^{N} \alpha_i P_{ng(i)}(\cdot), \text{with} \sum_{i=0}^{N} \alpha_i = 1$$

Within the (binary) decision tree, each leaf node C represents a set of samples, where a sample S is a $1:N$ pair from the aligned lexicon. A question about the context of the sample is associated with each non-leaf node. Two child nodes represent the positive and negative cases to this question. A sample can then be classified by traversing the tree and answering the questions at each node until the leaf node $C(S)$ is reached. For this node, we calculate

$$P_{dt}(\boldsymbol{\varphi}|C) = \frac{N(\boldsymbol{\varphi},C)}{N(C)},$$

where $N(\boldsymbol{\varphi},C)$ represents the number of samples in C producing $\boldsymbol{\varphi}$ and $N(C)$ the total number of samples in C. As training criterion, the entropy h is used:

$$-h := \sum_{S} logP_{dt}(\boldsymbol{\varphi}(S)|C(S))$$

The idea now is to find a tree which describes the training data best. We start with a trivial tree consisting of one node and grow the tree by splitting leaf-nodes using a question from a given set of questions. At each split, we select the leaf node and question which maximize the entropy.

Finally, the n-gram and decision tree model are log-linearly combined, whereas the interpolation parameter α is chosen empirically:

$$logP_{ngdt}(\boldsymbol{\varphi}|\boldsymbol{g}) = \alpha logP_{ng}(\boldsymbol{\varphi}|\boldsymbol{g}) + (1-\alpha)logP_{dt}(\boldsymbol{\varphi}|\boldsymbol{g})$$

An overview about decision tree models for grapheme-to-phoneme conversion is also given in [6, 7].

2.2. IBM Joint ME n-Gram Model (ibm)

In [8], a joint n-gram model is used to tackle the G2P task, very similar to the n-gram model presented in the previous section (cf. Eq. 1). The training schedule is somewhat different though. First, a unigram model is trained on the graphones with the conventional Baum-Welch EM algorithm. For all following iterations, Viterbi EM is applied increasing n by one. Features are added to the model for all n-grams occurring in the Viterbi chunking of the training data. The training procedure is continued until convergence of the model, realigning the data in each iteration.

2.3. Dragon Joint n-Gram Model (dra)

Within the method proposed in [9], the alignment is determined in a preprocessing step. $N:1$ graphones are selected via an HMM mechanism. Starting form uniform distributions, maximum likelihood (ML) phoneme model distributions are estimated using the Baum-Welch algorithm. After model training, Viterbi is used to find the single-best alignment. A concatenative unit refinement step is possible by joining the m highest-ranking graphone pairs sorted by bigram frequency. For the actual model training, the joint probability $Pr(\boldsymbol{g},\boldsymbol{\varphi})$ is calculated in the following way:

$$Pr(\boldsymbol{g},\boldsymbol{\varphi}) = \sum_{\boldsymbol{q}\in S(\boldsymbol{g},\boldsymbol{\varphi})} Pr(\boldsymbol{q})$$
$$= \max_{\boldsymbol{q}\in S(\boldsymbol{g},\boldsymbol{\varphi})} \prod_{i=1}^{|\boldsymbol{q}|} p(q_i|q_{i-1},\ldots,q_1)$$

Here $S(\boldsymbol{g},\boldsymbol{\varphi})$ denotes the set of all co-segmentations of $\boldsymbol{g}$ and $\boldsymbol{\varphi}$. The decoding is finally done using a best-first multi-stack algorithm, which is an approximation to the joint probability. For more details, the reader is referred to the original publication.

2.4. Sequitur (seq)

Within the Sequitur G2P toolkit[1] [7], again a joint n-gram model is used. The graphonemic model $p(q_i|q_{i-1},\ldots,q_1)$ is estimated using ML EM training on an existing pronunciation dictionary. For the possibly non-unique segmentation into graphones, a maximum approximation is applied.

2.5. Phonetisaurus (ps)

Phonetisaurus utilizes weighted finite-state transducers for decoding as a representation of a graphone-based n-gram LM trained on data aligned by an advanced $M:M$ alignment algorithm [10]. This alignment is provided by a variant of the EM algorithm [11]. The n-gram model is trained using the MIT LM toolkit [12], in which Kneser-Ney discounting with interpolation is used for smoothing. Decoding is done using OpenFST [13] with the following sequence of operations:

$$\text{nBest}(\pi(\text{oProj}(W \circ M)))$$

Here, W denotes the input FSA, which is a graph representation of the input word including grapheme clusters seen in training as alternative paths. M is the n-gram model encoded as FST. The W and M FSTs are composed and a projection onto the output symbols is performed. π denotes the removal of unwanted symbols like epsilons or sentence begin/end markers.

3. Experimental Results

For a fair comparison of the various methods, we performed model training and optimization on exactly the same data. As performance measures, we use phoneme error rate (PER) and word error rate (WER) to assess the quality of the G2P models on text data. They are defined as the Levenshtein distance divided by the number of phonemes in the reference pronunciation resp. as the fraction of words containing at least one error. Scoring is done using the NIST sclite scoring toolkit [14]. In some of the used lexica, more than one reference pronunciation is given per word. A hypothesis is considered correct if it equals one of the given reference variants. For the ASR experiments, we report the usual word error rates. Note that for both, G2P and ASR, the WER measure is used, but on a different level. Whereas the errors for G2P are measured on phoneme and word level, the corresponding levels in ASR would be word and sentence level.

[1] http://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html

Table 1: G2P results on the English Celex task. n denotes the best performing context length.

method	n	PER/WER[%]		
		train	dev	test
ngdt	5	0.8 / 3.8	3.4 / 15.8	3.4 / 15.8
ibm	9	1.4 / 6.4	3.9 / 17.3	3.9 / 17.4
dra	5	0.3 / 1.5	3.5 / 15.5	3.4 / 15.2
seq	8	0.1 / 0.6	2.7 / 11.8	2.5 / 11.4
ps	6	1.0 / 6.2	3.6 / 18.0	3.4 / 16.7
ROVER		0.2 / 1.0	2.6 / 12.0	2.5 / 11.6

Table 2: Statistics for lexica in various languages.

language	# symbols		∅ word length		# unique
	source	target	source	target	words
English	76	50	8.8	7.6	283k
German	65	48	13.3	12.1	436k
French	72	38	10.3	7.7	319k
Italian	63	38	10.2	10.1	252k
Dutch	66	45	11.5	9.9	347k

3.1. Results on Text Data

First experiments have been performed on the publicly available, mid-sized, British-English Celex corpus[15]. It consists of 40k training words, and a dev and test set with 5k and 15k words respectively. 26 different graphemes and 58 phonemes are used. The results are presented in Tab. 1.

The seq model performs best while the other models are in the same range, except the IBM model is somewhat worse. It is also interesting to see that the optimal n-gram context size varies between 5 and 9. For the ngdt approach, the decision tree context lengths are 4 and 1 on grapheme and phoneme level respectively. We did also apply ROVER system combination [16], but there are no improvements. One reason might be that the models are too similar (they all rely on graphone-based n-grams). W.r.t. alignment restrictions, seq performed best when using 0-1 symbols on grapheme and phoneme side per graphone, while for ps it was 0-2 symbols each.

To get an idea how these methods perform on large state-of-the-art tasks, we built some study sets for various languages. The statistics for the data sets are given in Tab. 2. The average number of pronunciations per word is < 1.09 for all lexica. We randomly selected 5% from the data for dev and test set each and kept these sets fixed for all experiments. Note that the number of source symbols denotes symbols seen at least ten times in the training data, including upper and lower case letters, accented characters as well as punctuation marks. In total, 151 characters are included at least once per language.

In Tab. 3, the performance on the English data set is presented for the various methods. Here, sequitur achieves the best performance, but the distance to phonetisaurus is only marginal. For this task, ROVER gives a small improvement. Note that this problem is harder than usual, since roughly five times as many characters are allowed within the grapheme input, since the systems should be able to produce a phoneme sequence for more or less any input character sequence (end user customization). In Tab. 4, we present the error rates on the test set for the remaining languages. Overall, seq and ps achieved the best results. The baseline system will be used for the ASR experiments in the next section and is included in the tables for reference.

Table 3: G2P results on the English data set. n denotes the best performing context length.

method	n	PER/WER[%]		
		train	dev	test
baseline	3	6.1 / 26.9	9.2 / 38.8	9.0 / 38.2
ngdt	5	2.2 / 10.0	5.6 / 24.8	5.7 / 24.9
ibm	14	2.6 / 11.8	5.3 / 22.4	5.2 / 22.7
dra	6	0.4 / 2.0	5.6 / 23.7	5.7 / 24.9
seq	9	0.3 / 1.7	4.8 / 21.0	4.9 / 21.4
ps	8	0.4 / 1.3	4.8 / 20.6	5.0 / 21.4
ROVER		0.4 / 1.8	4.6 / 20.3	4.6 / 20.6

Table 4: G2P results on the remaining test data sets (PER/WER[%]).

method	German	French	Italian	Dutch
baseline	6.1 / 47.8	2.7 / 13.3	3.0 / 21.8	2.8 / 18.4
ngdt	4.4 / 36.6	1.5 / 7.2	1.9 / 14.4	1.5 / 9.9
ibm	3.2 / 25.3	1.5 / 7.4	1.8 / 13.0	2.0 / 11.8
dra	3.6 / 27.1	1.9 / 9.1	1.8 / 12.1	1.5 / 9.9
seq	3.3 / 25.5	1.3 / 6.5	1.5 / 10.6	1.3 / 8.4
ps	3.1 / 23.7	1.3 / 6.4	1.6 / 11.0	1.2 / 7.8
ROVER	3.0 / 23.8	1.3 / 6.3	1.5 / 10.7	1.2 / 7.8

3.2. ASR Results

We used some of the optimized G2P systems on several study sets to assess the effect on the ASR word error rate. Besides the G2P system used to generate pronunciations for words where no manual transcription is available, the system setup is exactly the same per language. We apply a typical, contemporary two-pass ASR system comprising speaker independent models and adaptation on utterance level. Since we are mostly interested in measuring the difference in recognition quality across various G2P models, the exact ASR system setup is not so important to interpret the experimental results.

The statistics for the test sets are given in Tab. 5. As one can see, the ratio of words with automatically generated pronunciations is small on the test sets. Thus, it might be difficult to measure the effect of improved G2P modeling. Additionally, we were particularly interested how the methods studied fared against an established baseline system with accuracies $> 90\%$. (cf. Tab. 3 and 4).

In Tab. 6, we report recognition results for the baseline, the

Table 5: Data statistics for the various study sets. G2P ratio denotes the portion of the test set for which the pronunciation(s) in the recognition lexicon have been generated with the G2P model.

	total data[h]	running words	vocab size	G2P ratio [%]	OOV ratio [%]
English	157.3	629k	39k	0.44	1.00
German	184.5	719k	81k	0.80	3.22
French	177.7	807k	126k	1.66	1.81
Italian	176.7	627k	54k	0.31	2.51
Dutch	31.3	231k	20k	1.24	2.35

Table 6: ASR recognition results on various LVCSR study sets. Lines starting with "+OOV" denote systems where all OOVs from the test set have been added to the vocabulary (cheating experiment).

	WER [%] on test set (WER on segs containing words with prons by G2P model | WER on segs with OOVs)			
	baseline	dra	seq	seq-nBest
English	20.32 (28.74 | 47.19)	20.32 (28.26 | 47.10)	20.33 (28.38 | 47.21)	20.29 (26.95 | 46.95)
+OOV	19.84 (28.44 | 36.40)	19.78 (27.94 | 34.87)	19.78 (28.08 | 34.56)	19.68 (26.69 | 32.36)
German	21.30 (30.26 | 29.97)	21.31 (30.32 | 29.93)	21.29 (30.17 | 29.94)	21.40 (29.62 | 30.09)
+OOV	19.69 (28.85 | 24.80)	19.66 (28.84 | 24.71)	19.58 (28.59 | 24.49)	19.53 (28.02 | 24.03)
French	27.93 (36.68 | 43.86)	27.88 (36.36 | 43.78)	27.92 (36.63 | 43.87)	27.87 (36.05 | 43.70)
+OOV	27.31 (36.22 | 38.89)	27.24 (35.92 | 38.52)	27.24 (36.06 | 38.35)	27.16 (35.56 | 37.77)
Italian	24.79 (33.58 | 41.23)	24.77 (33.17 | 41.18)	24.77 (33.12 | 41.18)	24.78 (32.46 | 41.21)
+OOV	23.29 (32.52 | 33.64)	23.22 (32.06 | 33.14)	23.28 (31.99 | 33.43)	23.11 (31.17 | 32.42)
Dutch	28.17 (32.30 | 35.95)	28.01 (32.01 | 35.83)	28.08 (32.07 | 35.90)	28.05 (31.71 | 35.93)
+OOV	27.06 (31.58 | 32.00)	26.89 (31.20 | 31.68)	26.88 (31.27 | 31.55)	26.88 (30.83 | 31.18)

dra and the seq G2P system. Besides the WER on the complete test sets, we also report the WER on two possibly overlapping subsets. The first one contains utterances with at least one word with automatically generated pronunciation and the second one utterances with at least one OOV. As already suspected, the WER does not change much since only few words are affected. Since our goal was to measure the quality of the G2P systems, we did a cheating experiment and added all OOVs to the ASR vocabulary and used the respective G2P system to generate their pronunciations. These results can be found in the same table (lines beginning with "+OOV"). Now, the effect is stronger, especially when looking at the OOV segments only. For example, for English, when the baseline G2P model is used, the WER on OOV segments is 36.40 and drops to 34.56 when the optimized seq model is used instead. We did one run of experiments where we added nBest pronunciation variants for all words where no manual pronunciation is available instead of single-best. Here, we used the seq model and generated up to three pronunciations, depending on the overall posterior probability mass of the generated variants, which has been thresholded to < 0.75. When looking at the cheating experiment, the variants seem to help. But the overall (non-cheating) error rates do not necessarily improve, since more confusion may be introduced (e.g. for German).

4. Conclusions

We have presented a comparison of various state-of-the-art G2P models on large tasks on both, G2P accuracy and their effect on ASR performance within contemporary LVCSR systems on challenging tasks. The Sequitur and Phonetisaurus tools seem to outperform the other tested methods. With a cheating experiment where OOVs have been added, it could be shown that improved G2P modeling can be measured within ASR systems even over a highly competitive baseline. In any case, improving G2P is always beneficial for end user customization.

5. References

[1] T. Deselaers, S. Hasan, O. Bender, and H. Ney, "A Deep Learning Approach to Machine Transliteration," in *Proceedings of the EACL 2009 Workshop on Statistical Machine Translation*, Athens, Greece, Mar. 2009, pp. 233–241.

[2] S. Hahn, M. Dinarelli, C. Raymond, F. Lefevre, P. Lehnen, R. De Mori, A. Moschitti, H. Ney, and G. Riccardi, "Comparing Stochastic Approaches to Spoken Language Understanding in Multiple Languages," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 6, pp. 1569–1583, 2010.

[3] S. Deligne, F. Yvon, and F. Bimbot, "Variable-Length Sequence Matching for Phonetic Transcription Using Joint Multigrams," in *Proceeding of the Fourth European Conference on Speech Communication and Technology (EUROSPEECH)*, Madrid, Spain, Sep. 1995, pp. 2243–2246.

[4] S. Jiampojamarn and G. Kondrak, "Online Discriminative Training for Grapheme-to-Phoneme Conversion," in *Proceedings of ISCA Interspeech*, Brighton, U.K., Sep. 2009, pp. 1303–1306.

[5] R. Kneser, "Grapheme-to-Phoneme Study," Philips Speech Processing, Germany, Tech. Rep. WYT-P4091/00002, November 2000.

[6] A. Kienappel and R. Kneser, "Designing Very Compact Decision Trees for Grapheme-to-Phoneme Transcription," in *Interspeech*, Aalborg, Denmark, Sep. 2001, pp. 1911–1914.

[7] M. Bisani and H. Ney, "Joint-Sequence Models for Grapheme-to-Phoneme Conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, May 2008.

[8] S. F. Chen, "Conditional and Joint Models for Grapheme-to-Phoneme Conversion," in *European Conference on Speech Communication and Technology (EUROSPEECH)*, Geneva, Switzerland, Sep. 2003, pp. 933–936.

[9] P. Vozila, J. Adams, Y. Lobacheva, and T. Ryan, "Grapheme to Phoneme Conversion and Dictionary Verification Using Graphonemes," in *European Conference on Speech Communication and Technology (EUROSPEECH)*, Geneva, Switzerland, Sep. 2003, pp. 2469–2472.

[10] J. Novak, "Phonetisaurus: A WFST-driven Phoneticizer," 2011, http://code.google.com/p/phonetisaurus/.

[11] S. Jiampojamarn, G. Kondrak, and T. Sherif, "Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion," in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics NAACL-HLT)*, Rochester, NY, USA, 2007, pp. 372–379.

[12] P. H. Bo-June and J. Glass, "Iterative Language Model Estimation: Efficient Data Structure & Algorithms," in *Interspeech*, Brisbane, Australia, Sep. 2008, pp. 841–844.

[13] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: a general and efficient weighted finite-state transducer library," Prague, Czech Republic, Jul. 2007, pp. 11–23.

[14] NIST, "Speech Recognition Scoring Toolkit (SCTK)," http://www.nist.gov/speech/tools/.

[15] R. Baayen, R. Piepenbrock, and L. Gulikers, "Celex2," 1996.

[16] J. Fiscus, "A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER)," Santa Barbara, CA, Dec. 1997, pp. 347 – 354.