

English grapheme-to-phoneme conversion and evaluation

Uwe Reichel¹ Hartmut R. Pfitzinger² Horst-Udo Hain³

¹Institute of Phonetics and Speech Processing (IPS), Munich, Germany

²Institute of Phonetics and Digital Speech Processing (IPDS), Kiel, Germany

³Siemens AG, Corporate Technology, Munich, Germany

reichelu@phonetik.uni-muenchen.de, hpt@ipds.uni-kiel.de,
horst-udo.hain@siemens.com

ABSTRACT

In this study three English grapheme-to-phoneme (G2P) conversion models are presented and comparatively evaluated with respect to phoneme output, syllabification, and word stress location. The models are given by 1) artificial neural networks (SIE), 2) C4.5 decision trees (IPS), and 3) an information gain tree (Pfitzinger). Overall performance ranged from 27.25% to 33.4% word error rate. Model SIE significantly outperformed the other two.

1 Introduction

Grapheme-to-phoneme (G2P) conversion approaches can roughly be divided into rule-based [1] and data-driven procedures. While rule-based approaches profit from the direct application of linguistic knowledge, data driven methods have the advantage of inexpensive trainability as well as a low or zero degree of language dependence. Data-driven approaches include amongst others neural networks [2], instance based learning [3], tree classifiers [4] [5], and Hidden Markov models [6].

The models described in this paper follow the data-driven tradition. While two of them are purely data-driven (SIE, Pfitzinger), one also incorporates (automatically derived) linguistic knowledge (IPS). In addition to the phoneme sequence the models also predict word stress location and syllable boundaries.

2 Data

The data for training and testing was taken from a lexicon containing UK-Sampa transcriptions of English words, that had been developed at Siemens AG, Corporate Technology, Munich. After the removal of non-standard and foreign language

words and the unification of transcription variants 48460 entries remained for this study. Orthography and transcriptions were aligned separately for each model.

Corrections Potentially wrong reference transcriptions were identified by mismatches of at least two of this study’s models after an initial training pass. Whenever needed (in about 400 cases) the references were corrected.

3 G2P Models

3.1 Model SIE

This approach uses two neural networks, the first for the generation of the phoneme sequence including the syllable boundaries, and the second for the determination of the word stress for this phoneme sequence (cf. Figure 1).

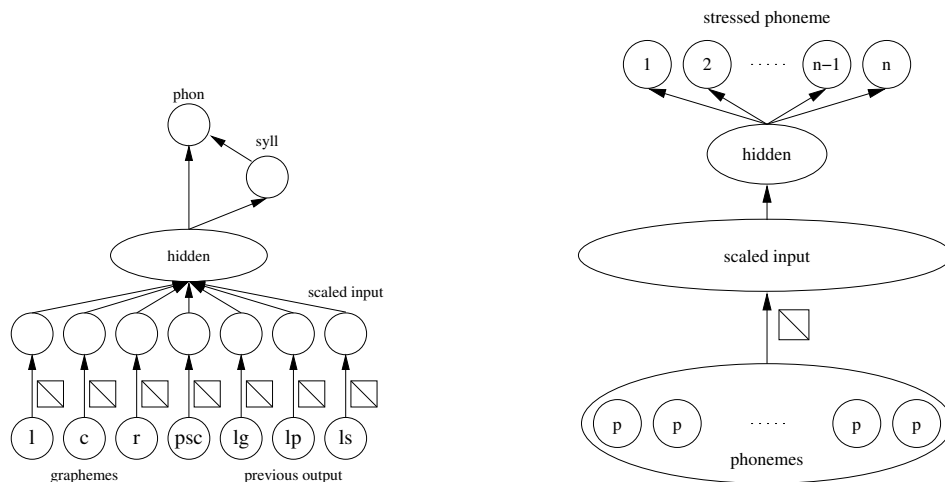


Figure 1: **Left:** Network for G2P conversion and syllabification. **Right:** Network for word stress location.

Phonemes and Syllable Boundaries Task for this network is to convert the grapheme sequence of the word into a phoneme sequence including the syllable boundaries. The input information is split into two parts, the grapheme input on the left and the previous phoneme output on the right hand site.

The grapheme input window consists of the centre grapheme (c) as well as the four preceding (l) and the four subsequent (r) graphemes. The input information about the previous phoneme contains a flag whether the previous phoneme was

a syllable core or not (psc) and the values for grouping (lg), phoneme (lp) and syllable (ls) of the previous phoneme. The grouping is the number of graphemes that were used to generate the output phoneme. If for instance the graphemes <tch> are converted to phoneme /tʃ/, then the grouping has the value three. Input nodes with no value (left context and previous output in the beginning of the word or right context at the end of the word) are set to zero.

There are two output nodes, one for the phoneme (phon) and one for the syllable boundary (syll) which is a flag that marks whether there occurs a syllable boundary behind the output phoneme or not. The phoneme node has as many nodes as there exist phoneme-grouping combinations. For instance for phoneme /tʃ/ there exist three output nodes /tʃ_1/ (generated by <t>, <c>), /tʃ_2/ (generated by <ch>, <cc>) and /tʃ_3/ (generated by <tch>, <che>). The output node with the highest value determines the generated phoneme-grouping combination.

Word Stress This network determines the position of the word stress for the given phoneme sequence. The input layer contains the first ten phonemes of the word, the nodes for the missing phonemes are again set to zero. The output layer has the same number of nodes as there are phonemes in the input layer. The output node with the highest value determines the position of the stress within the phoneme sequence. If for instance the third node has the highest value, then the third phoneme of the word carries the stress.

Network Architecture and Training Both networks have a special layer between input and hidden layer, which is called *scaled input*. The input layer and this scaled input are connected by a diagonal matrix, which means that each input node is connected by exactly one weight with its appropriate node of the scaled input. On this connection the so called weight decay [7] is applied. By this algorithm weights are driven to zero unless they are really necessary for the solution. This is similar to a pruning of the input layer and makes the network more insensitive to input noise [8].

3.2 Model IPS

The model IPS consists of four C4.5 decision trees [9] for orthographic and phonological syllabification, for G2P conversion, and for word stress assignment. Some of the features used by these trees are provided by a Viterbi part-of-speech tagger and an automatic morphologic segmentation [10]. Figure 2 shows the information flow between the modules.

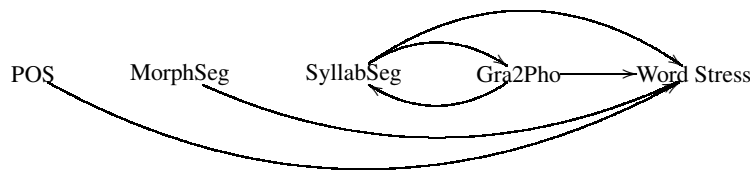


Figure 1: *Information flow within the IPS model.*

Figure 2: *Information flow within the IPS model.*

The modules for G2P conversion and syllabification are described in greater detail in [10].

Syllable segmentation In orthographic syllabification for each letter l_i is decided whether a syllable boundary follows or not by taking into account a letter window of length 7 centered on l_i . This information is used for subsequent G2P conversion.

In phonological syllabification for each phoneme p_i derived by the G2P module (see below) the decision whether or not to let it be followed by a boundary depends on the phonemes in a window of length 7 centered on p_i , the sonority relation between p_i , p_{i-1} , and p_{i+1} , as well as on the place and mode of articulation of p_i , p_{i-1} , and p_{i+1} .

G2P conversion For conversion of letter l_i to phoneme p_i , which can also be the “empty phoneme” or a phoneme cluster to permit n -to- n -mappings, the following features are used, which are partly based on the output of the orthographic syllabification:

- Current letter g_i and surrounding letters $g_{i-n} \dots g_{i+n}$ (n from 1 to 4)
- Syllable boundary +/- following
- Position within the syllable (head, nucleus, coda)
- Type of the current syllable (onset/null onset, open/closed)
- Relative position of g_i within the word
- Phoneme history of length 3

Word stress assignment The position of the stressed syllable (one stress location per simplex word or per compound part) is identified relatively to the word end. Following features based on the outputs of all the other modules are used:

- Part of speech
- Prefix and suffix string
- Number of syllables
- Weight of the last 4 syllables: reduced, light, heavy
- Reverse position of the syllable within the word

3.3 Model Pfitzinger

The basic principle of the Pfitzinger grapheme-to-phoneme converter is 1) to align the training lexicon, i.e. assigning all letters of each word to the corresponding phonemes of the transcriptions, 2) to transform it into an IG-tree (information gain tree) which finally is compressed without loss, optionally pruned, and extended with best-guess leaves, and 3) to use the reduced IG-tree to transduce orthography into its phonemic transcription.

It is mainly inspired by the description of a language-independent and data-oriented approach of Daelemans and van den Bosch [5]. The main components and improvements which characterise the present system will be described as follows.

The Aligner uses two DP-passes: 1) for each phoneme of a word a triangular window with a width of 5 letters is centered at that letter whose relative position in the word corresponds to the relative position of the phoneme in the transcription in order to spread the probability of co-occurrence also to adjacent letters. 2) The resulting co-occurrence matrix is converted into probabilities and used by a Dynamic Programming algorithm to find the most likely alignment for each lexicon entry. 3) A corrected co-occurrence matrix is estimated which contains only little noise and probable two- or three-phoneme-symbols. 4) It is used by a second DP-step to estimate the final alignments.

The Generator 1) creates a tree for each letter containing letters (alternately from the right and the left context) as nodes, and phonemes or multi-phoneme-symbols as leaves. All letters of all words in the training lexicon are added to the trees. 2) The trees are minimised by replacing all branches with only one type of

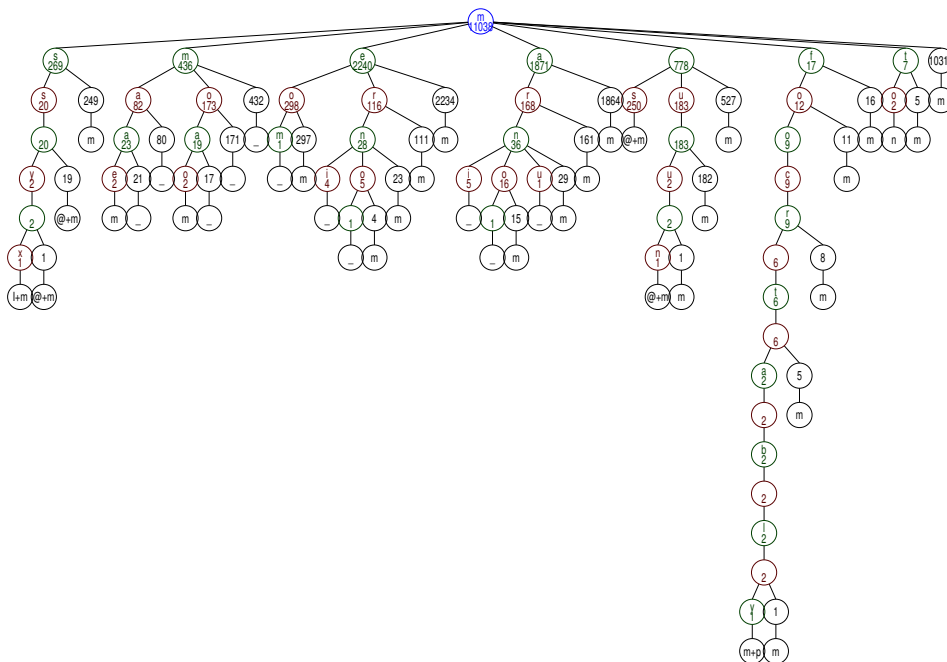


Figure 3: IG-tree of the letter *m*. The case of *m* becoming an /*m+p*/ is because the word *comfortable* is transcribed as /*k* "V *m* - *f* @ - *t* @ - *b* @ *l*/ in the training lexicon while mistakenly *comfortably* as /*k* "V *m* *p* - *f* @ - *t* @ - *b* *l* I/.

leaf by this leaf. 3) Optionally, branches with a very small number of visits during the construction are pruned followed by a second minimisation step. 4) For each branch which does not comprise the whole set of possible letters a best-guess leaf is added to achieve generalisability. The leaf gets the most frequent phoneme of all sibling branches. As an example Fig. 3 shows the tree for the letter *m*.

The Translator consists of a linking of the reduced IG-tree and an efficient tree traversing algorithm that traces along the IG-tree nodes and leaves to convert words into their corresponding phonemic transcriptions.

Features When pruning is skipped, all training data ambiguity is resolved and unambiguous training data is predicted correctly by 100%. The full 10-fold training and evaluation procedure is finished in 70 seconds on a 3 GHz Core2Duo CPU, and 1,5 Mio. letters per second are transduced with 87,000 internal nodes representing an English lexicon of 48,500 entries.

4 Evaluation

4.1 Procedure

A 10-fold cross validation was carried out. As standard evaluation measures the word error rates were calculated for the complete transcriptions WER, for the transcriptions without word stress WER(-stress), without syllabification WER(-syl), and without both WER(-stress,-syl).

4.2 Results

The word error rates introduced above are presented in table 1 and Figure 4. Model Pfitzinger is not yet capable of predicting syllable boundaries, therefore no performance is reported in the corresponding cells.

Table 1: Word error rates (medians).

	SIE		IPS		Pfitzinger	
	train	test	train	test	train	test
WER	13.33	27.25	18.68	33.4	–	–
WER (-stress)	12.08	24.07	11.13	26.74	–	–
WER (-syl)	12.51	26.17	16.46	31.34	0	34.63
WER (-stress,-syl)	11.25	22.93	8.36	24.23	0	23.53

Applied on the test data model SIE performs significantly better than the other models in all considered word error rates (Wilcoxon test of paired samples, $p < 0.01$). Model Pfitzinger significantly outperforms model IPS in raw grapheme-to-phoneme conversion, while it is significantly outperformed by model IPS in predicting also word stress location (Wilcoxon test of paired samples, $p < 0.05$).

5 Discussion

Adaptation to training data As can be seen in Table 1 all models show highly different performances on training and test data. Since up to now the Pfitzinger IG-trees are not pruned, they are capable to achieve 100% accuracy for the training data as long as no homographs are contained.

The high performance differences of the other models could be partly explained by the fact that while in training they are provided by clean data, in application they face the problem of error accumulation (see below).

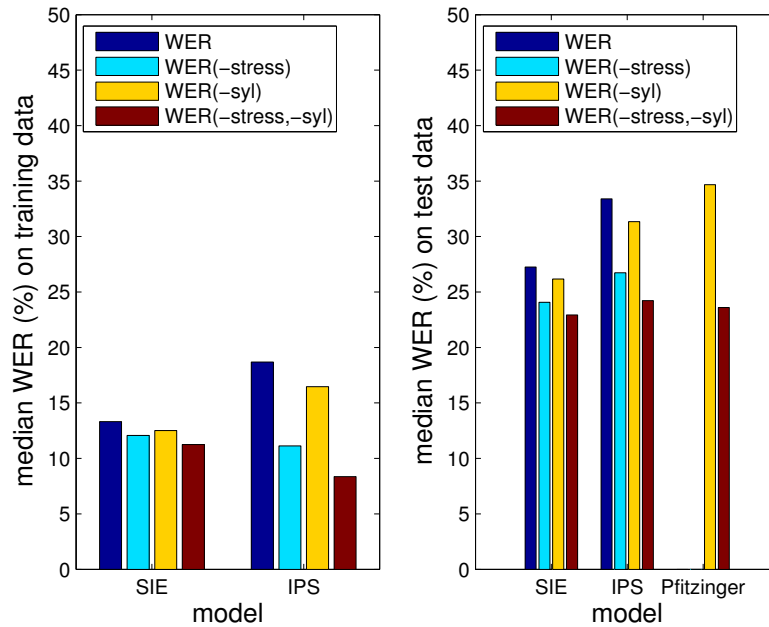


Figure 4: Median model performances on training (left) and test data (right). Bars from left to right: a) Overall performance, b) without word stress, c) without syllabification, d) without stress and syllabification. Zero training data word error for model Pfitzinger due to lack of pruning.

Error accumulation Model Pfitzinger follows a “predicting all at once”-strategy for phonemes and word stress and therefore is not confronted with the problem of error accumulation. In contrast, the other models partly distribute these tasks over several modules, some of which operating on other module’s sometimes defective outputs. This circumstance might explain why the performance differences on training and test data are especially high for SIE and IPS, when word stress localisation is concerned, which is carried out by the final module.

Feature selection A comparison of the differences between WER and WER(-stress) for models SIE and IPS shows that linguistically motivated higher-level features (e.g. syllable weight, morpheme class) as been used within IPS do not guarantee a better performance compared to the low-level features (e.g. letters, phonemes) used by SIE. In this study they even lead to poorer performances.

This finding might be partly caused by the following factors:

- The automatic extraction of higher-level features is error-prone.
- Extracting higher-level features is more generally affected by error accumulation than extracting low-level features.
- Relying on few high-level features instead of many low-level features is more vulnerable to noise, since in the former setting a wrongly extracted feature has higher fatal impact on the classification result, than in the latter.

Byproducts The Pfitzinger IG-tree could be used to identify idiosyncrasies and therefore potential errors in the lexicon, which are represented by long branches. This capability qualifies this model as a tool for lexicon corrections.

6 Acknowledgements

We would like to thank Ute Ziegenhain and Gabriele Bakenecker from Siemens AG, Corporate Technology, Munich, for providing us with the pronunciation lexicon, and all the members of the ECESS for fruitful discussions.

References

- [1] D. Stock, “P-TRA – Eine Programmiersprache zur phonetischen Transkription,” in *Beitrge zur Angewandten und Experimentellen Phonetik*, W. Hess and W. Sendlmeier, Eds. Stuttgart: Franz Steiner Verlag, 1992.
- [2] S. T.J. and R. C.R., “Parallel networks that learn to pronounce English text,” in *Complex Systems*, 1987, vol. 1.
- [3] A. van den Bosch and W. Daelemans, “Data-oriented methods for grapheme-to-phoneme conversion,” ITK Research Report, Tilburg, Tech. Rep. 42, 1993.
- [4] O. Andersen and P. Dalsgaard, “Multi-lingual testing of a self-learning approach to phonemic transcription of orthography,” in *Proceedings Eurospeech*, 1995, pp. 1117–1120.
- [5] W. Daelemans and A. van den Bosch, “Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion,” in *Progress in Speech Synthesis*. New York: Springer, 1997, pp. 77–89.

- [6] S. Parfitt and S. R.A., “A bidirectional model of English pronunciation,” in *Proceedings Eurospeech*, Genova, Italy, 1991, pp. 801–804.
- [7] A. Krogh and J. Hertz, “A Simple Weight Decay Can Improve Generalisation,” in *Advances in Neural Information Processing Systems*, J. Moody, S. Hanson, and R. Lippmann, Eds., San Mateo CA, 1995, vol. 4, pp. 950–957.
- [8] H.-U. Hain and H. Zimmermann, “A multi-lingual system for the determination of phonetic word stress using soft feature selection by neural networks,” in *Proc. Fourth ISCA ITRW on Speech Synthesis (SSW4)*, 2001.
- [9] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann, 1993.
- [10] U. Reichel and H. Pfitzinger, “Text preprocessing for speech synthesis,” in *Proc. TC-Star Speech to Speech Translation Workshop*, Barcelona, Spain, 2006, pp. 207–212.