Phoneme-to-Grapheme Mapping for Spoken Inquiries to the Semantic Web

Axel Horndasch[*†], *Elmar Nöth*[*], *Anton Batliner*[*], *Volker Warnke*[†]

[*] Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, Germany
[†] Sympalog Voice Solutions GmbH, Erlangen, Germany
{horndasch,noeth,batliner}@informatik.uni-erlangen.de, {horndasch,warnke}@sympalog.de

Abstract

Automatic methods for grapheme-to-phoneme (G2P) and phoneme-to-grapheme (P2G) conversion have become very popular in recent years. Their performance has improved considerably, while at the same time these developments required less input from expert lexicographers. Continuing in this tradition we will present in this paper a data-driven, language-independent approach called MASSIVE[1] with which it is possible to create efficient online modules for automatic symbol mapping. Our framework is solely based on statistical methods for training and run-time and has been optimized for P2G conversion in the context of spoken inquiries to the Semantic Web, an issue researched in the SmartWeb project[2]. MASSIVE systems can be trained using a pronunciation lexicon, the output of a phone recognizer or any other suitable set of corresponding symbol strings. Successful tests have been performed on German and English data sets.

Index Terms: phoneme-to-grapheme/sound-to-letter conversion, out-of-vocabulary words, Semantic Web.

1. Introduction

To be confronted with the task of converting symbol sequences from one representation to another is quite common in the field of speech processing. Automatic grapheme-to-phoneme conversion (G2P), also referred to as spelling-to-pronunciation or letter-to-sound conversion, plays an especially significant role when generating spoken language from free text input. The conversion of phonemes to graphemes (P2G, also known as pronunciation-to-spelling or sound-to-letter conversion) on the other hand is needed less frequently, but it can be necessary when dealing with out-of-vocabulary words (OOVs) in spoken dialogue systems. If for example the output of a speech recognizer for an OOV hypothesis is the underlying sequence of phonemic sub-word units, as suggested in [1], it has to be converted to a sequence of letters if the dialogue module expects graphemic input.

The goal of the SmartWeb project [2] is to build a question-answering system that uses the Semantic Web as its knowledge base. Because its multi-modal user interface provides for spoken language inquiries, the need to handle out-of-vocabulary words is a crucial point concerning speech recognition. This setup also affects the requirements regarding the P2G module, which should for example be able to produce n-best graphemic transcriptions for OOV words which can be used for an intelligent search.

Consider for example a request like "Show me information about the soccer player *Burruchaga*!" Having no other information source, most people would probably misspell the player's name, even if they heard the correct pronunciation `/burUtSa:ga/`[3]. With the Semantic Web as a backup and a number of spelling alternatives the task of finding the right answer to the user's request becomes a lot easier.

Our research has benefited greatly from earlier work on (bi-directional) G2P and P2G conversion, e. g. [3, 4, 5]. The main focus of this paper is to present a fully automatic approach for phoneme-to-grapheme mapping in the context of open vocabularies. The approach is based on probabilistic methods for setting up real-time systems offering utmost flexibility with respect to the input data. In the next section we shortly describe how we combine speech recognition and the Semantic Web for question answering in SmartWeb. The algorithms we use at run-time and for creating new conversion systems within the MASSIVE framework are introduced in section 3. In section 4 we present the results of our experiments and how they can be compared to the results of other studies. We conclude with an outlook on how the system could be enhanced in the future.

2. Speech recognition and the Semantic Web

Since questions to SmartWeb are not restricted concerning the domain, it is important to rely on a combination of strategies for handling unknown words in the speech input. The basic approach is to use a hybrid speech recognizer[4], into which the detection and processing of OOV words is integrated. For every OOV hypothesis a semantic category is also returned, a method first mentioned in [6].

P2G result	Rank	Google	Google + S
burachaga	1	36	4
boorachaga	2	0	0
burruchaga	3	108 000	627
bourachaga	4	0	0
buruchaga	5	785	31

Table 1: *Google hits for grapheme hypotheses converted from the phoneme string* `/burUtSa:ga/`.

Table 1 gives an idea on how the Semantic Web can help with respect to unknown words. In the first column the top five grapheme hypotheses for `/burUtSa:ga/` returned by a

[1] The name was chosen because with the approach *Mapping Arbitrary Symbol Sets Is Very Easy*.

[2] http://www.smartweb-project.de

[3] We will use SAMPA (http://www.phon.ucl.ac.uk/home/sampa/) for all phonemic representations of words throughout this paper.

[4] With *hybrid* we refer to the fact that the recognizer was trained to produce words and sub-word units in its output.

P2G system created with MASSIVE are shown. The third column (labeled *Google*) contains the number of Google hits for the grapheme hypothesis. In the fourth column (labeled *Google + S*) the number of Google hits for the hypothesis and the search words *soccer player* are displayed, simulating semantic context deduced from the inquiry. The correct transcription `burruchaga`[5] in the third row is the unchallenged winner in both categories.

Having motivated the creation of n-best results, we want to point out some other requirements that were important to us while developing the MASSIVE framework. Apart from real-time processing we wanted full flexibility with respect to other sub-word units that might be used in future versions of the hybrid speech recognizer. Also the issue of creating a P2G system for an English language demonstrator for SmartWeb had to be taken into account.

3. The MASSIVE system

The basic idea behind MASSIVE is to create a sequence of symbol pairs, e. g. of phonemes and graphemes, which are derived from the alignment of corresponding representations in different alphabets. The symbol pairs, which can be looked upon as words, are then used for training a statistical language model; given a new string during run-time, the conversion is done by splitting the input into single symbols, creating probable pairs of input and output symbols and doing a graph search.

lexicon entry
`/IgzA:mpl,/` → `example`

initial alignment
`I_e g_x z_a A:_m m_p p_l l,_e`

final alignment
`I_e g_x z_NIL A:_a m_m p_p l,_l NIL_e`

Figure 1: *An example for iterative phoneme-grapheme alignment. The _-symbol separates entries from input and output alphabet.*

It should be obvious that the quality of the alignment has a major influence on the performance of the resulting conversion system. Another problem that has to be dealt with are insertions and deletions which are indicated using the `NIL`-symbol in figure 1. MASSIVE attacks these issues during the training phase by an iterative alignment procedure and the clustering of output symbols.

3.1. Iterative alignment of symbol sequences

Automatically aligning phonemic and graphemic representations of a word is not trivial. Even if the number of symbols is the same, the corresponding sounds and letters may be located at different positions leading to insertions and deletions (as is the case in the example in figure 1). Additionally, the most probable phonetic counterpart of a letter depends very much on the position within a word and the language.

To generate correct sequences of symbol pairs, MASSIVE uses a naive initial alignment as a starting point. If possible, similarities between input and output symbols can be taken into account at that stage (e. g. the same character encoding). After that an EM-like algorithm is applied:

While old and new alignment differ

1. Compute distances between input and output symbol based on the frequency of the according symbol pair in the last alignment (*expectation step*)
2. Align all corresponding symbol sequences again with respect to the new distances using dynamic programming (*maximization step*)

Similar ways to align phoneme and grapheme strings have been reported in other studies (e. g. in [7]), however most approaches use manually compiled grapheme or phoneme clusters to make a one-to-one mapping possible or the seeding for an initial alignment is not automatic.

3.2. Automatic clustering of output symbols

The result of an alignment process always contains insertion or deletion patterns caused by symbol clusters in one representation of the word being mapped onto fewer symbols in the other representation. For example the letter cluster `ough` can be mapped onto a single symbol in SAMPA in different ways (e. g. `/O:/` as in *thought* or `/u:/` as in *through*). To think of all possible clusters in advance is almost impossible, even if abbreviations and neologisms like `Xmas` are filtered out. For example the alignment of the symbol strings `asthma` and `/{sm@/` results in the very unintuitive grapheme cluster `sth`.

While there are many grapheme clusters which can be mapped to a single phoneme, correspondences of single letters to more than one phoneme also exist. For example the German letter `z` has the affricate `/ts/` as the most probable phonemic equivalent, a similar case is the letter `x` which is usually pronounced `/ks/`. A typical phoneme cluster in the English language is `/ju:/` for the letter `u`.

From these examples it becomes clear that manual clustering is not easy; it has to be done separately for every language or, more generally speaking, for every different type of input. However, clustering can greatly facilitate the problem of modeling insertions in a conversion system, so that for MASSIVE an automatic approach is the only feasible option.

lexicon entry
`/A:gju:IN/` → `arguing`

alignment result
`A:_a NIL_r g_g j_NIL u:_u I_i N_n NIL_g`

clustering (first step)
`A:_a NIL_r g_g j_NIL u:_u I_i` **N_ng**

clustering (second step)
A:_ar `g_g j_NIL u:_u I_i N_ng`

Figure 2: *Clustering output symbols*

Clustering symbols or, to be more precise, augmenting symbol sets with clusters also has a negative effect: the segmentation of strings becomes ambiguous. Consider the word `firsthand`, in which the grapheme clusters `sth` and `th` can be found. In this case the letters `s`, `t` and `h` all contribute to the pronunciation of the word[6] and should not be regarded as a cluster. To circumvent

[5] The soccer player Jorge Burruchaga won the World Cup in 1986 with Argentina scoring the decisive goal in the final against Germany.

[6] The CMU Pronunciation Dictionary actually contains a second entry for `firsthand` which does not have a `/t/` in its phonemic representation.

the segmentation problem we decided against clustering symbols in the input alphabet. In figure 2 for example the final result of the clustering process still contains a NIL-symbol on the right hand side of a symbol pair (j_NIL). As a consequence we have to allow deletions to occur during run-time.

Our clustering algorithm, with which the example in figure 2 was generated, looks like this:

While there are NIL-symbols on the input side of symbol pairs (the left hand side)

1. Count all n-grams with at least one symbol pair containing a NIL-symbol in the input part and one neighboring pair with a non-NIL input symbol
2. Replace the most frequent n-gram with a new symbol pair by merging input and output symbols from all pairs in the n-gram (the NIL symbols are of course removed from the input part)

It has to be noted that even if the alignment result is of high quality, this process generates errors which lead to incorrect phoneme-grapheme correspondences. An example is the word *ricocheted* for which the sequence of symbol pairs r_r I_i k_coch @_e S_t eI_e d_d is generated by the alignment of rIk@SeId and ricocheted. Unexpected clusters can be caused by words from another language, abbreviations or erroneous correspondences in the input. For example during our experiments with the German wordforms from the CELEX Lexical Database [8] we found over 2500 errors[7] simply by taking a closer look at the grapheme clusters.

3.3. Training n-gram models with symbol pairs

The n-gram probabilities for the symbol pairs can be estimated with standard maximum likelihood techniques using the results obtained from alignment and clustering. The formal definition of the probability of a sequence of symbol pairs can be written as

$$P(o_1\_q_1, \ldots, o_T\_q_T) = \prod_{t=1}^{T} P(o_t\_q_t | o_1\_q_1, \ldots, o_{t-1}\_q_{t-1}).$$

The equation demonstrates the close relationship of our approach to the concept of joint n-gram models. Joint n-gram models are a way of dealing with ambiguities by augmenting words (or as in our case symbols) with extra information [9].

At run-time we face the task of mapping a sequence of symbols $\boldsymbol{o}$ to the corresponding $\boldsymbol{q}$ in the output alphabet. This is done with a simple beam search algorithm:

Split input string into single symbols
For the current input symbol

1. Generate all possible symbol pairs
2. Compute conditional probability for symbol pair
3. Take best sequences according to beam width
4. Next symbol

Remove all input symbols and return output string(s)

[7] The lion's share of the errors were words containing the letter sequence schst which was in most cases transcribed as /St/ instead of /Sst/.

4. Evaluation

In the experiments for this paper we used three different sets of data: the German version of the CELEX Lexical Database, the CMU Pronunciation Dictionary [10] and the output of a phone recognizer trained on the EVAR corpus [11]. Cross validation test results include the mean value and standard deviation (enclosed in parentheses). Phoneme (p-acc), grapheme (g-acc) and word accuracies (w-acc) were computed by subtracting the error rates (the sum of all deletions, insertions and substitutions divided by the number of all symbols/words) from 100%.

4.1. Experiments on the CELEX Lexical Database

For our tests on German data we took the SAMPA version of all wordforms (over 360000) contained in the CELEX Lexical Database (release 2, German version 2.5) and divided them up into equally large sets for 10-fold cross-validation. The set of phonemes consisted of 59 entries, including a lot of international symbols but no affricates, diphthongs or any other clusters. We trained a 5-gram language model with the set of phoneme-grapheme pairs we got from alignment and clustering.

P2G	g-acc [%]	w-acc [%]
aligned (naïv)	97.2 (0.1)	86.2 (0.2)
it. aligned	97.5 (0.1)	88.1 (0.2)
al. & clustered	99.6 (0.0)	96.1 (0.1)

Table 2: *phoneme-grapheme mapping results on CELEX (German wordforms)*

Table 2 shows the results of the phoneme-to-grapheme (P2G) task. It can be seen that even by just aligning phoneme and grapheme strings and training the language model, the results concerning the grapheme accuracy are already in the high nineties. This is mainly due to the large corpus size. However it can be seen that by iterative alignment and clustering the word accuracy can be improved drastically. The average number of phoneme-grapheme pairs extracted by the clustering procedure was 601.

Because MASSIVE can also be used for G2P conversion, we created a G2P system based on the German wordforms from CELEX. In our experiments, we were able to achieve an average phoneme accuracy of 99.6% and a word accuracy of 96.8%, with the same setup as for the P2G experiments.

4.2. Experiments on the CMU pronunciation dictionary

The setup for our tests on the CMU data is similar to the one used for CELEX. We split up the dictionary into ten parts consisting of about 12700 entries each for cross-validation tests. We filtered out non-alphanumeric symbols except for the apostrophe[8]. Table 3 shows grapheme and word accuracies for n-best hypotheses generated with the MASSIVE system for the P2G task. In [3] word and letter accuracies of 50.3% and 91.2% respectively are reported (the conditions described are very similar but only a single test set was used and no n-best output was presented). Our results come very close to that benchmark although, for efficiency reasons, we are doing without some of the concepts applied in [3] (e. g., no class-based smoothing of symbol pairs, no exhaustive search).

[8] The reason for this was to make our results comparable to the ones reported in [3].

P2G	g-acc [%]	w-acc [%]
best	88.7 (0.1)	50.0 (0.3)
2-best	93.0 (0.1)	65.1 (0.4)
3-best	94.7 (0.1)	72.4 (0.3)
4-best	95.7 (0.1)	76.9 (0.3)
5-best	96.4 (0.1)	80.0 (0.3)
6-best	96.8 (0.1)	82.2 (0.4)
7-best	97.1 (0.1)	84.0 (0.3)
8-best	97.4 (0.1)	85.3 (0.3)
9-best	97.6 (0.1)	86.4 (0.3)
10-best	97.8 (0.1)	87.2 (0.3)

Table 3: *n-best phoneme-grapheme mapping results for 10-fold cross-validation on the CMU Pronunciation Dictionary*

Considering the power of the Semantic Web, it is encouraging that in 87.2% of all cases the first 10 hypotheses contained the correct spelling. With a reordering of the hypotheses based on the method described in section 2, the chances of coming up with the correct spelling in first place are a lot better.

4.3. Experiments on the output of a phone recognizer

Since the motivation for MASSIVE was to come up with a component for phoneme-to-grapheme conversion in the framework of out-of-vocabulary speech recognition, we also experimented with the conversion of data produced by a phone recognizer. We took 10000 turns from the EVAR corpus for training and 2500 turns for testing a monophone recognizer (Mono_Rec). With this simple setup we obtained a 65% phone accuracy along with a 43% accuracy at the turn level (see table 4). In [11] a turn accuracy (turn-acc) of 72.8% on the same data with a word recognizer and a 4-gram category-based language model has been reported.

	Mono_Rec	P2G_Lex	P2G_Rec
swu-acc [%]	65.3	66.5	68.2
turn-acc [%]	43.2	38.5	47.6

Table 4: *Phoneme accuracy of the recognizer and grapheme accuracies after the P2G-conversion of the recognizer output using systems trained with a lexicon (P2G_Lex) and training material of the recognizer (P2G_Rec)*

For testing phoneme-to-grapheme conversion we generated two P2G modules. The first (P2G_Lex) was trained using the EVAR pronunciation lexicon. The set of corresponding phoneme and grapheme strings for the second module (P2G_Rec) was taken from the training material of the monophone recognizer. With both systems we came up with a better sub-word unit accuracy (swu-acc) after the P2G conversion. One of the reasons for this is, of course, that there were considerably fewer grapheme (29, including umlauts and the ß character) than phoneme classes (57).

More interesting is the fact that the system trained with data produced by the monophone recognizer does significantly better than the lexicon-trained component in terms of grapheme and turn accuracy (see tables 4). Apart from having more data to learn from, P2G_Rec can adjust to typical errors produced by the recognizer whereas P2G_Lex has only seen neat correspondences from the lexicon.

5. Conclusion

We presented a data-driven framework called MASSIVE which can be used to automatically generate phoneme-to-grapheme and grapheme-to-phoneme conversion systems. It is specially tailored for a scenario involving spoken inquiries to the Semantic Web by generating n-best graphemic transcriptions for out-of-vocabulary words. The approach is solely based on statistical methods, flexible towards the type and the language of the input and focused on run-time efficiency. Experiments on German and English test sets showed that MASSIVE is competitive w. r. t. approaches reported in the literature. To further improve the system, we are planning to integrate new ideas regarding the preprocessing of the training data, for example by doing alternating alignment and clustering steps.

6. Acknowledgements

This research is being supported by the German Federal Ministry of Education and Research (BMBF) in the framework of the SmartWeb project under Grant 01IMD01A. The responsibility for the contents of this study lies with the authors.

7. References

[1] Bazzi, I. and Glass, J. R., "Modeling Out-Of-Vocabulary Words for Robust Speech Recognition", in *Proc. ICSLP '00*, pp 401-404, Beijing, China, 2000.

[2] Wahlster, W., "Smartweb: Mobile Applications of the Semantic Web", in *GI Jahrestagung '04*, pp 26–27, 2004.

[3] Galescu, L. and Allen, J., "Bi-directional Conversion Between Graphemes and Phonemes Using a Joint N-gram Model", in *Proc. 4th ISCA Workshop on Speech Synthesis*, Pitlochry, Scotland, 2001.

[4] Decadt, B., Duchateau, J., Daelemans, W. and Wambacq P. "Transcription of Out-of-Vocabulary Words in Large Vocabulary Speech Recognition Based on Phoneme-to-Grapheme Conversion", in *Proc. ICASSP '02*, Orlando, FL, USA, 2002.

[5] Rentzepopoulos, P. A. and Kokkinakis, "Efficient Multilingual Phoneme-to-Grapheme Conversion Based on HMM", in *Computational linguistics*, 22(3), pp 341–376, 1996.

[6] Gallwitz, F., Nöth, E. and Niemann, H., "A Category Based Approach for Recognition of Out-of-Vocabulary Words", in *Proc. ICSLP '96*, pp 228–231, Philadelphia, PA, USA, 1996.

[7] Pagel, V., Lenzo, K. and Black A., "Letter to Sound Rules For Accented Lexicon Compression", in *Proc. ICSLP '98*, Sidney, Australia, 1998.

[8] Baayen, R. H., Piepenbrock, R. and Gulikers, L., "The CELEX Lexical Database (Release 2) [CD-ROM]", 1995.

[9] Galescu, L. and Ringger, E., "Augmenting Words With Linguistic Information For N-Gram Language Models", in *Computational linguistics*, 22(3), pp 341–376, 1996.

[10] Weide, R., "The CMU Pronunciation Dictionary, release 0.6", Carnegie Mellon Mellon University, 1998.

[11] Stemmer, G., "Modeling Variability in Speech Recognition", PhD thesis, Universität Erlangen-Nürnberg, Lehrstuhl für Mustererkennung, Germany, 2005.