Memory-Based Phoneme-to-Grapheme Conversion

A Method for Dealing with Out-of-Vocabulary Items in Speech Recognition

Bart Decadt, Jacques Duchateau, Walter Daelemans and Patrick Wambacq

CNTS Language Technology Group, University of Antwerp
ESAT-PSI Speech Group, K.U. Leuven

Abstract

In this paper, we describe a method to enhance the readability of out-of-vocabulary items (OOVs) in the textual output in a large vocabulary continuous speech recognition system. The basic idea is to indicate uncertain words in the transcriptions and replace them with phoneme recognition results that are post-processed using a phoneme-to-grapheme (P2G) converter.

We concentrate on the final step, P2G conversion: we show that the phoneme recognition results can be reasonably reliably transcribed orthographically using machine learning techniques. More specifically, (i) we present experimental results of a machine learning approach to P2G conversion, and compare these results with an estimation of the upper and lower baseline performance, (ii) we give an error analysis and list some examples of the converter's output, (iii) we investigate spelling correction as post-processing of the orthographic transcriptions, and (iv) we report on the interaction of the P2G converter with a speech recognizer.

1 Introduction

One of the major problems in speech recognition is the reliable recognition of words not present in the speech recognizer's vocabulary (out-of-vocabulary items, OOVs). Current speech recognition technology makes use of (among other information sources) a restricted pronunciation lexicon (typically 40k words) to produce word graphs (lattices of possible sequences of words detected in the input) from which the most likely sequence is chosen. This approach cannot handle words not present in the restricted lexicon: when an OOV occurs in the input speech, the speech recognizer cannot map that part of the input to a word in its lexicon, and maps it to a similar sounding sequence of words from the lexicon, which can make the output difficult to read. In ESAT's speech recognizer (Duchateau, Demuynck and Van Compernolle 1998, Demuynck, Duchateau, Van Compernolle and Wambacq 2000), for example, the word *gespreksonderwerp* (*topic of conversation*) is not in the lexicon: the speech recognizer maps it to the words *gesprek zonder werk* (*conversation without work*).

A possible solution to this problem is to detect these OOVs using confidence measures provided by the speech recognizer, produce a phoneme string for them using a phoneme recognizer, and finally use a phoneme-to-grapheme (P2G) converter to find a likely orthographic transcription of the OOV (see also: (Decadt, Duchateau, Daelemans and Wambacq 2001) and (Decadt, Duchateau, Daelemans and Wambacq 2002)).

In the next section, we outline the basic system architecture and introduce the

Proceedings CLIN 2001, 47-61, Rodopi, 2002

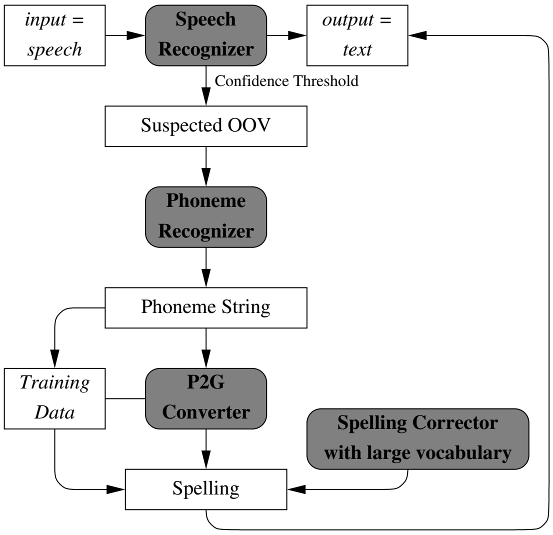

Figure 1: A speech recognizer enhanced with a phoneme-to-grapheme converter

machine learning technique we used to build the P2G converter. In section 3, we present the results of experiments with the P2G converter on the output of ESAT's phoneme recognizer, and estimate an upper and lower baseline performance on the P2G conversion task. We analyze the errors in the converter's output in section 4, and investigate spelling correction of the output as a possible post-processing step, in section 5. Finally, in section 6, we report on the interaction of the P2G converter with the speech recognizer.

2 Memory-based Phoneme-to-Grapheme Conversion

Basic system architecture. Figure 1 is a schematic representation of the basic architecture of a large vocabulary speech recognizer enhanced with a P2G converter. First, we extend the ESAT speech recognizer with confidence measures for detecting possible OOVs in the input speech. The suspected OOVs serve as input for the ESAT phoneme recognizer, which produces a phoneme string for the OOV.[1] This phoneme string then serves as input for a P2G converter, which turns the phonemes into a grapheme string. Finally, spelling correction with a large vocabulary is used as a post-processing step. The resulting grapheme string is then put in its place in the output text from the speech recognizer.

[1]The phoneme recognizer mentioned is not a separate system: for phoneme recognition, we use the speech recognizer with a vocabulary of 40 phonemes, instead of using it with a large vocabulary of 40k words. The context-dependent acoustic modeling and the statistical model of phoneme sequences (5-gram) were estimated on a dataset containing six hours of speech read aloud (different from the CGN recording (see section 3) used as training data).

In this paper, we concentrate on the P2G converter: we will see that P2G conversion is a fairly easy task with perfect input - however, the phoneme strings generated by the phoneme recognizer are not free from errors: the typical error rate for a phoneme recognizer is $\sim$25% (the sum of deletions, substitutions and insertions of phonemes). The motivation for the experiments reported here is our hypothesis that machine learning techniques can adapt to the peculiarities of the errors made by the phoneme recognizer, and can provide the necessary robustness and accuracy to the P2G conversion task when provided with sufficient training data (pairs of words and corresponding output of the phoneme recognizer).

Machine learning method. Our P2G converter is constructed with TIMBL, a memory-based learning implementation. Memory-based learning is based on the hypothesis that in domains like language processing, where relatively few regularities compete with many sub-regularities and exceptions, a *lazy* form of learning (keeping in memory all examples and using similarity-based reasoning on all examples at classification time) is superior to an *eager* learning approach (extracting rules or other abstractions from the examples and using these to handle new cases) (Daelemans, van den Bosch and Zavrel 1999). Furthermore, the results of research on a similar task (grapheme-to-phoneme conversion (Daelemans and van den Bosch 1996, van den Bosch and Daelemans 1998, Busser, Daelemans and van den Bosch 1999, Hoste, Gillis and Daelemans 2000)), suggest that memory-based learning may be very well suited for our task, P2G conversion.

TIMBL is a software package for memory-based learning implementing a wide range of algorithms, weighting metrics, and other parameters. It can take as input patterns (or instances) of feature values with a corresponding class symbol (supervised, example-based learning). During the learning phase, TIMBL stores all instances in memory and collects statistical data about these instances. To evaluate the performance of TIMBL on a task, a test set containing previously unseen instances is used: TIMBL predicts the class of these new instances by comparing them with the instances from the training set. The new instance gets the same label as the most similar instance(s) from the training set. We will describe here only the algorithms which we used in our experiments, for a full description of the implementation of all available algorithms and metrics, we refer to (Daelemans, Zavrel, van der Sloot and van den Bosch 2001).

The basic similarity between two instances is computed using an *overlap metric*. In the case of our symbolic, nominal data (phonemes as features), this means that similarity between two patterns is the number of features for which the two patterns have the same value. Obviously, this would in general give bad results as not all features are equally relevant for solving a particular task. We use an information-theoretic approach (*information gain* in its form normalized for number of values per feature; i.e. *gain ratio*, see (Quinlan 1993)) to weigh the relevance of the different features. We will call this algorithm IB1-IG, introduced in (Daelemans and van den Bosch 1992). Another factor of importance in memory-based learning is the number of neighbors that is taken into account to extrapolate from (the parameter *k*). Finally, we have used in our experiments the IGTREE algorithm (Daelemans, van den Bosch and Weijters 1997), a decision tree based heuristic approximation of memory-based learning which is more efficient than IB1-IG.

Data preprocessing. In the machine learning set-up we chose, each phoneme of each word is represented with its surrounding context as an instance or pattern that has to be classified with the grapheme corresponding with that phoneme (there are as many patterns to be classified as there are phonemes). This implies that to work properly, the grapheme and the phoneme strings for each word should be of equal length. As the phoneme strings are rarely as long as their corresponding grapheme strings, they have to be *aligned*. Most grapheme strings are longer: we use *compound graphemes* to shorten the grapheme strings. For example, in the Dutch word *slaap* (*sleep*), with pronunciation //, we replace the graphemes *aa* with the *compound grapheme A*:

slaap → s l A p → / /

In the reverse case, a shorter grapheme string, we insert the null symbol '-' in that string. An example is the alignment of the Dutch word *taxi* with its phoneme string //:

taxi → t a x - i → / /

We insert these null symbols with the *Dynamic Programming* algorithm (also known as *Dynamic Time Warping*) (Wagner and Fischer 1974, Kondrak 2000). Given two strings to be aligned, this algorithm computes an alignment cost for each pair of symbols in the two strings, and stores this cost in a matrix. When the cost for each pair is computed, the algorithm searches for the least expensive way through the matrix.

The context of a phoneme consists of its preceding and following phonemes: with a context-size of three phonemes, for example, the Dutch word *kast* (*cupboard*), pronounced //, would be represented as the four instances below:

Left context			Focus	Right context			Class
=	=	=					k
=	=					=	a
=					=	=	s
				=	=	=	t

The last symbol in a pattern (i.e., the class to be output) always indicates the orthographic representation of the phoneme in focus, which here occurs in fourth position, while the other positions represent phonemes in the context of the focus phoneme, with the symbol '=' indicating a word boundary.

3 Experimental Results

We experimented with the P2G converter on two datasets: the first dataset was the Dutch word list (174k words with their pronunciation) from CELEX (Baayen,

1. Upper Baseline (experiments with CELEX)	IB1-G $k=1$	IB1-G $k=3$	IB1-G $k=5$	IGTREE
graph. level acc.	**99.1%**	98.9%	98.9%	99.0%
word level acc.	**91.4%**	90.2%	89.7%	91.2%

2. Lower Baseline (exp. with phon. rec. output)		Basic statistical approach
COMPLETE DATASET	graph. level acc.	70.5%
	word level acc.	30.0%
OOVS IN DATASET	graph. level acc.	60.2%
	word level acc.	3.0%

3. P2G Converter Performance (exp. with phon. rec. output)		IB1-G $k=1$	IB1-G $k=3$	IB1-G $k=5$	IGTREE
COMPLETE DATASET	graph. level acc.	76.2	77.3	**77.4**	76.4
	word level acc.	46.4	**46.5**	**46.5**	46.3
OOVS IN DATASET	graph. level acc.	58.1	62.3	**63.0**	59.1
	word level acc.	6.2	6.7	**6.9**	6.1

Table 1: Results (accuracy in % at word and grapheme level) of the experiments with the P2G converter

Piepenbrock and van Rijn 1993); the second dataset was made by running the ESAT phoneme recognizer on a recording (129k words, with an orthographic transcription) from the Corpus Gesproken Nederlands (CGN, *Spoken Dutch Corpus*).[2] The first one was used to estimate the upper baseline performance on the P2G conversion task, whereas the second one was used to estimate the lower baseline performance, and to train and test the P2G converter. In both datasets, we aligned the phoneme and grapheme strings as described in the previous section.

The phoneme strings produced by the phoneme recognizer contain three kinds of errors: substitutions, insertions and deletions (the total error rate is ∼25%). Substitutions make the classification task more difficult: for each phoneme, there will be more possible graphemes (or *class labels*). Insertions are not that difficult to handle: the P2G converter has to convert the inserted phonemes to *empty* graphemes. However, the deletions are problematic: the architecture of our P2G converter requires a one-to-one correspondence between phonemes and graphemes, and as we are not able to predict where deletions occur in an unseen phoneme string, the P2G converter can never convert a phoneme string with deletions to a completely correct word – there will be graphemes missing. In the CGN dataset, 27% of the words is recognized with deletions: even if the P2G converter

[2]The CGN project is sponsored by the Dutch NWO and the Flemish IWT, see `http://lands.let.kun.nl/cgn/ehome.htm`.

is able to convert all phonemes correctly, the maximum word level accuracy would not be higher than 63%.

3.1 Estimation of Upper and Lower Baseline Performance

Upper baseline performance. To estimate an upper baseline for the performance on the P2G conversion task, we used CELEX because the phoneme strings in this lexicon are free from errors (they do not contain substitutions, insertions and deletions of phonemes as in the phoneme recognizer's output). We trained and tested the P2G converter with *ten-fold cross-validation* (10CV, the dataset is split in 10 parts, and ten experiments are conducted with each part as test set while the remaining nine parts serve as training set) for various parameter settings for TIMBL: IB1-IG with k = 1, 3, and 5, and IGTREE. The context of a phoneme in focus position consisted of its three preceding and following phonemes. The results of this experiment are presented in the first part of Table 1: the best scoring parameter setting is IB1-IG with $k = 1$, resulting in 99.1% grapheme level accuracy and, more importantly, 91.4% word level accuracy. We see that P2G conversion is almost an easy task if the phoneme strings are free from errors.

Lower baseline performance. To estimate a lower baseline, we used the output of the phoneme recognizer on the CGN recording with a basic statistical approach to the P2G conversion task: we simply convert a phoneme to the most frequent grapheme for that phoneme. If the phoneme //, for example, corresponds in 78% of the cases with the grapheme *p* and in only 22% with *b*, then we always convert // to *p*. This experiment was also done with 10CV: probabilities were computed on nine parts of the dataset, and tested on the remaining part. The result is presented in the second part of Table 1: 70.5% at grapheme level, and 30.0% at word level for all words in the CGN dataset. As the OOVs are tagged in this dataset, we can also give figures for these words: 60.2% at grapheme level, and 3.0% at word level.

In the basic statistical approach, adaptation to the peculiarities of the phoneme recognizer errors is not possible, because context is not taken into account and each phoneme has only one possible grapheme. The P2G converter, on the other hand, takes the previous and following phonemes into account, and can give multiple graphemes for one particular phoneme: if the converter adapts to the errors, it should score better.

3.2 Performance of the Phoneme-to-grapheme Converter

To test the performance of the P2G converter, we trained and tested the converter on the CGN data with 10CV for the same parameter settings of TIMBL as in the experiment with the CELEX data. The context of a phoneme was also the same: its three preceding and following phonemes. The results are presented in the third part of Table 1: the best scoring algorithm is IB1-IG with $k = 5$, resulting in 77.4% grapheme level accuracy and 46.5% word level accuracy for the complete dataset.

The conversion accuracy for the OOVs in the CGN dataset is much lower: 63.0% grapheme level and 6.9% word level accuracy.

Both for the complete dataset and for the OOVs only, at grapheme and at word level, the P2G converter with TIMBL scores better than the statistical baseline method, though the difference in performance is more outspoken in the results for the complete dataset. This indicates that there are probably few regularities for P2G conversion in the OOVs.

4 Error Analysis

Ambiguous phonemes. Examining the output of the P2G converter when trained and tested on CELEX data, we learn that, when the phoneme strings are free from deletions, substitutions and insertions, most errors are due to ambiguous phonemes. We distinguish three types of ambiguous phonemes: the most frequent type contains phonemes that have different possible spelling forms, and the spelling form belonging to a particular word is conventional (there are no contextual cues which can decide on the spelling form needed). Some examples of this type are listed in Table 2.

AMBIGUOUS PHONEME	PHONEMIC REPRESENTATION	P2G CONVERTER	CORRECT CONVERSION
// can be	//	incleding	inkleding
k or *c*	//	voetbalkompetitie	voetbalcompetitie
// can be	//	bijbelsitaten	bijbelcitaten
c or *z*	//	censatiebladen	sensatiebladen
// can be	//	elektrolitisch	elektrolytisch
i or *y*	//	fyle	file
// can be	//	ziektigedrag	ziektegedrag
e or *i*	//	oorlogscrises	oorlogscrisis
// can be	//	zeefouna	zeefauna
au or *ou*	//	triplexhaut	triplexhout
// can be	//	zenuwleider	zenuwlijder
ei or *ij*	//	uitwijdt	uitweidt

Table 2: Some examples of ambiguous phonemes due to convention

Words in which assimilation processes are at work may introduce ambiguity in phonemes which are otherwise not ambiguous. The phoneme //, for example, usually has to be converted to the grapheme *m*, but when a // follows, it is possible (though not necessary) to convert it to *n*. In some cases, there is not enough contextual evidence to decide on one of the two alternatives. In Table 3 are some examples of this second type of ambiguous phonemes.

Words (or parts of words) with the same pronunciation but a different spelling, can also result in incorrect predictions. The P2G converter predicts *ladikant* as

KIND OF ASSIMILATION	PHONEMIC REPRESENTATION	P2G CONVERTER	CORRECT SPELLING
// → // before //	//	eembaansweg	eenbaansweg
// → // before //	//	stugbreken	stukbreken
// → // at word-end	//	rotatietijt	rotatietijd
// → // at word-end	//	lop	lob
// → // after //	//	dagsuster	dagzuster
// → // after //	//	praktijkfakken	praktijkvakken

Table 3: Some examples of ambiguous phonemes due to assimilation

the spelling for the phoneme string // (*ledikant*), because CELEX contains a lot of words beginning with *lady-* (*ladyshave*, *ladykiller*, *ladylike*, …). The same goes for the string // (*lieders*), which is converted to *leaders*, because *lieder(s)* and *leader(s)* are pronounced in the same way. Errors of this kind are not very frequent, though.

Some ambiguous words or phonemes can never be classified correctly by including only previous and following phonemes in the context: morphological or syntactic cues are needed to resolve the ambiguity. A typical example is the Dutch verb *worden* (to become), which is pronounced // in the first, second and third person singular (present tense) but is spelled differently: *word* in the first person, and *wordt* in the second and third person. Without morphological or syntactic cues, the P2G converter can never predict the correct spelling. Some examples are listed in Table 4.

PHONEMIC REPRESENTATION	P2G CONVERTER	CORRECT SPELLING
//	bespied	bespiedt
//	doodbloed	doodbloedt
//	onderscheidt	onderscheid
//	onderhoudt	onderhoud
//	afraat	afraadt

Table 4: Some examples of ambiguous phonemes for which morphological or syntactic cues are needed

Atypical spelling. Incorrect conversions are not always due to ambiguity: errors also occur in words which are spelled in a way that is not typically Dutch (mainly because these words come from other languages and were added to the Dutch vocabulary without adapting it to Dutch spelling conventions), like the words in Table 5.

PHONEMIC REPRESENTATION	P2G CONVERTER	CORRECT SPELLING
//	rokuille	rocaille
//	sykcurij	cichorei
//	peperclips	paperclips
//	tiekwondo	taekwondo
//	curasau	curacao
//	projectiems	projectteams
//	foyee	foyer
//	matiner	matinee
//	bazoeka	bazooka
//	gekroest	gecruist

Table 5: Some examples of Dutch words with an atypical spelling

Analysis of the OOVs. Looking at the OOVs, we see that the errors made by the P2G converter are not equally distributed over the OOVs: Figure 2 compares the expected number of errors in a word of length n (= word-length $\times$ average percentage of errors at grapheme level, i.e. 38.3% (100 - 61.7%)) with the observed average number of errors in the words of that length. The bars in Figure 2 depict the frequency of words with length n. We see that the two curves are not totally equal: short OOVs contain more errors than expected, while long OOVs have fewer errors. Only in words with length 10 to 15, the observed number of errors is more or less equal to the expected number. However, as the bars in Figure 2 illustrate, the shorter words, with a higher than expected number of errors, are more frequent than the longer words.

The low word-level accuracy for the OOVs does not mean that the P2G converter's output is not *readable*: incorrectly converted OOVs containing only one or two errors, could still give the reader a clear idea of what the correct word should be. Figure 3 shows that such OOVs are quite numerous: in the output from the experiment with the CGN data, we counted how many OOVs had a certain amount of errors per word, and put the averages in the chart in Figure 3. The bad news is that OOVs with 3, 4, 5 and even 6 incorrectly predicted graphemes per word - words which should be difficult to read - are as frequent as the *readable* ones.

5 Spelling Correction as a Post-processing Step

In the previous section, we noted that (i) short OOVs, on average, tend to have more errors than expected, and (ii) OOVs containing 4, 5 and even 6 errors are quite frequent. On the basis of these observations, we did not expect an enormous improvement from using a spelling corrector for post-processing. Assuming that only words with 1 or 2 errors have a reasonable chance of being corrected by a spelling corrector, the maximum increase in word level accuracy we may expect,

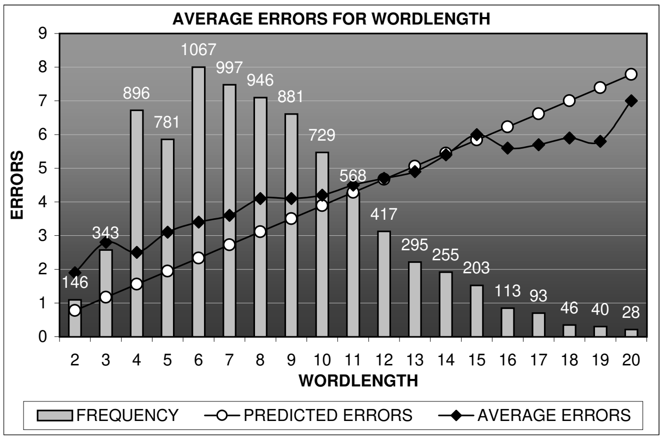

Figure 2: Average errors per word for word-length in the OOVs

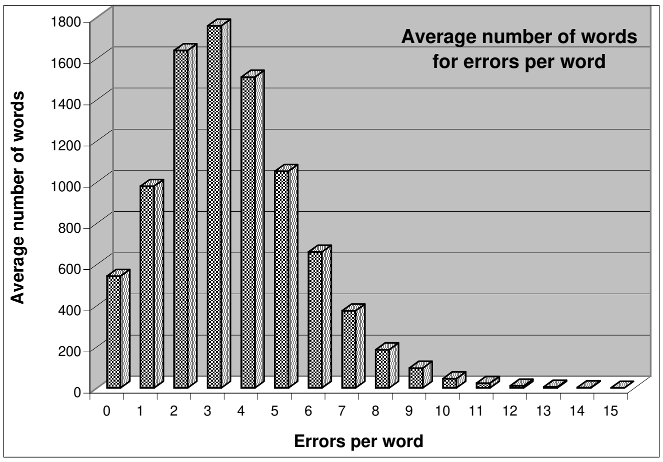

Figure 3: Number of words for errors per word in the OOVs

<table>
<tr><th colspan="2">Correctly converted words: 612 (6.9%)</th><th colspan="2">Incorrectly converted words: 8280 (93.1%)</th><th></th></tr>
<tr><td>marked as correct</td><td>487</td><td>marked as correct</td><td>1291</td><td></td></tr>
<tr><td>marked as incorrect with suggestions</td><td>109</td><td>marked as incorrect with suggestions</td><td>3018</td><td></td></tr>
<tr><td>marked as incorrect without suggestions</td><td>16</td><td>marked as incorrect without suggestions</td><td>3971</td><td></td></tr>
<tr><th colspan="2">Loss in accuracy (correct words marked as incorrect)</th><th colspan="2">Gain in accuracy (incorrect words with correct suggestion)</th><th>Total accuracy</th></tr>
<tr><td colspan="2" rowspan="3">-1.4%</td><td>considering only the 1st suggestion:</td><td>+2.4%</td><td>7.8%</td></tr>
<tr><td>considering the first 3 suggestions:</td><td>+4.1%</td><td>9.6%</td></tr>
<tr><td>considering all suggestions:</td><td>+4.8%</td><td>10.3%</td></tr>
</table>

Table 6: The result of spelling correction on the output of the P2G converter with IB1-IG and *k* set at 5

is ∼25%. However, this is still quite an improvement compared to the ±7% word level accuracy we have now.

For our experiments, we did not develop a spelling corrector specifically adapted to our task: we used *Ispell*, UNIX' spelling corrector. The Dutch lexicon for *Ispell* contains 114k words, and a list of affixes to form new words with. *Ispell* can be used in an automatic mode in which each input word is checked for correctness: each word receives a mark (correct or incorrect), and for the incorrect ones, *Ispell* gives a list of suggestions, if there are any.

In Table 6, we present the results of running *Ispell* on the list of P2G conversions for the OOVs in ESAT's dataset (obtained by running the P2G converter with IB1-IG and *k* set at 5). These results are only indicative: we could get better results if we use a spelling corrector specifically adapted to our task, with e.g. a lexicon containing more proper names.

It is clear that *Ispell* is not able to correct most of the words containing only one or two errors: the gain in word level accuracy is 2.4% (taking into account only the first suggestion) to 4.8% (considering all suggestions). Moreover, spelling correction also degrades performance: we lose 1.4% word level accuracy because of correctly converted words marked as incorrect by *Ispell*. This does not mean that spelling correction is useless as a post-processing step: gaining 4.8% in word level accuracy (resulting in 10.3%) means an improvement of ∼70% compared to our previous result, 6.9%. Furthermore, *Ispell* is able to mark 84.4% of the incorrectly converted words as incorrect. This information can be used in the final transcriptions of the speech recognizer enhanced with our P2G converter, e.g. it can be presented with color codes.

6 Interaction with the Speech Recognizer

To test how the P2G converter influences the performance of the speech recognizer, we trained the P2G converter with the CGN data (see section 3), and tested the whole system (as presented in Figure 1) with a separate test set of 3.6k words.

The vocabulary of the speech recognizer consists of 40k words: these are the most frequent words in newspaper texts for which a phonetic transcription was available in a pronunciation dictionary (this excludes proper names). With this lexicon, a 3.5% OOV rate was found on a test set. The speech recognizer uses a trigram language model, trained on newspaper texts. The cross word context dependent acoustic modeling is based on a phoneme set with 38 three state phonemes and one noise state. A global phonetic decision tree defines 575 tied states, which are modeled with in total 10k tied Gaussians. These numbers are rather small due to the size of the acoustic training database for Dutch, namely 6 hours of speech.

With the above lexicon and modeling, a 14.7% word error rate (WER) was found on the test set. This is higher than the typical error rate for speaker independent large vocabulary recognition due to the small acoustic models and the high OOV rate. For comparison, with a similar type of acoustic modeling we achieved a 7.3% WER on the well known Wall Street Journal (WSJ) recognition task for the November 92 evaluation test set (trigram, 20k word vocabulary, 1.9% OOV rate, 69 hours of acoustic training data).

For the P2G converter, we find 55.2% accuracy at the word level on this test set: this number is only the average accuracy over all test set words. On the 3.5% OOVs, a word level accuracy of only 7.9% is found: the OOVs are often long words, or a-typical for Dutch. The word level accuracy on the recognition errors (including the OOV words) is 19.2%, one of the reasons for this low accuracy is that difficulties in the acoustic data (for instance a bad pronunciation for a word) will result in errors in both the word recognizer and the phoneme recognizer.

At the time of writing, the confidence measures were not yet implemented - we have to make an estimate of the interaction with what we find in the literature: from (Kemp and Schaaf 1997), we know that, for a recognition task with a WER as in our experiments, the threshold in the confidence measure can be adjusted so that about 75% of the recognition errors are tagged as *uncertain word* (thus missing 25% of the errors), while tagging (wrongly) only 10% of the correctly recognized words.

If we suppose that the 75% tagged recognition errors will be converted with the 19.2% accuracy average for recognition errors, and the 10% tagged correct words with the accuracy average for correct words (which is 59.9%), then transcriptions in which all tagged words are re-written by the P2G converter will result in a slightly higher WER: about 16.0% instead of the 14.7% mentioned earlier.

But this does not mean that the resulting transcriptions are less readable than the original ones. Albeit only 19.2% of the wrongly recognized words is transcribed correctly by the P2G converter, 41.0% is transcribed with at most 1 error (counting each substituted, inserted or deleted letter as an error), and 62.6% is transcribed with at most 2 errors.

A lot of the words with only a few errors can be understood by a reader. Moreover the transcribed words often do not exist in Dutch, giving the reader a clear lead that that word is uncertain (the original transcription is a concatenation of existing words, known by the recognizer).

As examples we give the longest words that are wrongly recognized by the speech recognizer. They are compounds, and OOV words for the recognizer. The transcription by the speech recognizer and by the P2G converter is given.

programmaproducent	→ programma producent (speech rec.)
(program producer)	→ programaprodusent (P2G)
gespreksonderwerp	→ gesprek zonder werk (speech rec.)
(topic of conversation)	→ gespreksonberwerp (P2G)
speelgoedmitrailleur	→ speelgoed moet hier (speech rec.)
(toy machine gun)	→ spergoetnietrijer (P2G)

For the recognition errors, only the first word is readable (the speech recognizer did not recognize it as a compound but as two separate words). The other two words are nonsense: *gesprek zonder werk* means *conversation without work* and *speelgoed moet hier* means *toys must here*. The P2G converter's output is closer to the correct words and more readable: *programaprodusent* contains two errors and *gespreksonberwerp* only one. The last conversion, *spergoetnietrijer*, containing 9 errors, is an example of a loan word with a spelling atypical for Dutch. Due to the fact that both the phoneme recognizer and the P2G converter are trained to produce strings typical for Dutch, these words will always result in bad conversions.

Finally, spelling correction again proves to be useful as post-processing: it increases the word level accuracy for OOVs to 8.7%, for recognition errors to 20.9%, and to 60.1% on average over all words in the test-set. The word error rate of the speech recognizer combined with our P2G converter then drops from 16.0% to 15.4%. From the examples above, only the second word (*gespreksonderwerp*) could be corrected by *Ispell*.

7 Conclusion

In this paper we investigated the feasibility of P2G conversion for OOVs in speech recognition, and the ability of machine learning methods for this task to adapt to the errors produced by the phoneme recognizer. We have discussed the results of experiments in which TIMBL, a memory-based learner was used for this task. We saw that it can carry out this task almost perfectly with a clean dataset, i.e. presupposing perfect phoneme recognition. In that case, 91% correctly transcribed words is feasible, with errors mainly related to the conventional lexical aspects of Dutch spelling.

Using a dataset with phoneme strings generated by a phoneme recognizer that contains more or less 25% errors, we achieved a lower, but still reasonable result: 46% at word level on the entire dataset. However, performance on the OOVs in this dataset, in which we are especially interested, is only 7% at word level (about 60%

of the graphemes correctly predicted). Although at a very low level, this accuracy may still be useful because many of the orthographically transcribed words can be recognized easily. Furthermore, post-processing the grapheme strings with a spelling corrector, proves to be useful, increasing the word-level accuracy to 8%.

An important problem is that the phoneme recognizer deletes a lot of phonemes, which is a situation impossible to handle in the current architecture of the system. Although word level accuracy on the OOVs is not very high, we showed that TIMBL did learn to adapt to the errors of the phoneme recognizer to a certain extent. Even when in an integration with the speech recognizer the total WER increases, readability of the output can nevertheless be improved with this method.

We believe the spelling error correction post-processing can be made more reliable by using lexicons and correction strategies tuned to OOVs and tuned to the type of errors the P2G module makes. More work can also be done on optimization of feature selection and algorithm parameters for the learning task, and the approach should be further tested in combination with different types of confidence measures.

Also, both the 5-gram statistical phoneme sequence model in the phoneme recognizer and the P2G converter are trained on Dutch in general, not specifically on OOV words. It may be better to train on OOV words only, as the properties of OOV words (typically loan words or proper names) may differ from the properties of Dutch in general.

Another direction for further research is the use of a more sophisticated description of the phoneme recognizer result. At this moment, the input for the P2G converter consists of only one phoneme string for a word. This means an important loss of information which may be useful for the converter. The use of a phoneme graph, possibly including probabilities for the phonemes, could be a solution to this problem.

Acknowledgments

This research is funded by IWT in the STWW programme, project ATraNoS.[3] We would like to thank Erik Tjong Kim Sang and Véronique Hoste from the CNTS research group for their support during the development of the proposed system.

References

Baayen, R. H., Piepenbrock, R. and van Rijn, H.(1993), *The* CELEX *lexical data base on* CD-ROM, Linguistic Data Consortium, Philadelphia, PA.

Busser, B., Daelemans, W. and van den Bosch, A.(1999), Machine learning of word pronunciation: the case against abstraction, *Proceedings of EuroSpeech99*, Budapest, Hungary, pp. 2123–2126.

Daelemans, W. and van den Bosch, A.(1992), A neural network for hyphenation, *in* I. Aleksander and J. Taylor (eds), *Artificial Neural Networks 2: proceedings*

[3] ATraNoS is the acronym for the project's title *Automatic Transcription and Normalization of Speech*. The project's homepage is located at `http://atranos.esat.kuleuven.ac.be`.

of the International Conference on Artificial Neural Networks, Elsevier, Amsterdam, pp. 1647–1650.

Daelemans, W. and van den Bosch, A.(1996), Language-independent data-oriented grapheme-to-phoneme conversion, *in* J. P. H. V. Santen, R. W. Sproat, J. P. Olive and J. Hirschberg (eds), *Progress in Speech Processing*, Springer-Verlag, Berlin, pp. 77–89.

Daelemans, W., van den Bosch, A. and Weijters, A.(1997), IGTree: using trees for compression and classification in lazy learning algorithms, *Artificial Intelligence Review* **11**, 407–423.

Daelemans, W., van den Bosch, A. and Zavrel, J.(1999), Forgetting exceptions is harmful in language learning, *Machine Learning, Special issue on Natural Language Learning* **34**, 11–41.

Daelemans, W., Zavrel, J., van der Sloot, K. and van den Bosch, A.(2001), TiMBL: Tilburg memory based learner, version 4.0, reference guide, *ILK Technical Report 01-04*, Tilburg University. Available from: `http://ilk.kub.nl`.

Decadt, B., Duchateau, J., Daelemans, W. and Wambacq, P.(2001), Phoneme-to-grapheme conversion for out-of-vocabulary words in large vocabulary speech recognition, *Proceedings of ASRU*, IEEE, Madonna di Campiglio.

Decadt, B., Duchateau, J., Daelemans, W. and Wambacq, P.(2002), Transcription of out-of-vocabulary words in large vocabulary speech recognition based on phoneme-to-grapheme conversion, *Proceedings of ICASSP*, IEEE, Orlando, Florida. To appear.

Demuynck, K., Duchateau, J., Van Compernolle, D. and Wambacq, P.(2000), An efficient search space representation for large vocabulary continuous speech recognition, *Speech Communication* **30**(1), 37–53.

Duchateau, J., Demuynck, K. and Van Compernolle, D.(1998), Fast and accurate acoustic modeling with semi-continuous HMMs, *Speech Communication* **24**(1), 5–17.

Hoste, V., Gillis, S. and Daelemans, W.(2000), Machine learning for modeling Dutch pronunciation variation, *in* P. Monachesi (ed.), *CLIN 1999. Selected papers from the tenth CLIN meeting*, pp. 73–83.

Kemp, T. and Schaaf, T.(1997), Estimating confidence using word lattices, *Proceedings of EuroSpeech97, vol. II*, Rhodes, Greece, pp. 827–830.

Kondrak, G.(2000), A new algorithm for the alignment of phonetic sequences, *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, NAACL, Seattle, pp. 288–295.

Quinlan, J. R.(1993), C4.5*: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

van den Bosch, A. and Daelemans, W.(1998), Do not forget: Full memory in memory-based learning of word pronunciation, *in* D. Powers (ed.), *Proceedings of NeMLaP3/CoNLL98*, Sydney, Australia, pp. 195–204.

Wagner, R. A. and Fischer, M. J.(1974), The string-to-string correction problem, *Journal of the Association for Computing Machinery* **21**(1), 168–173.