COMBINING OPEN VOCABULARY RECOGNITION AND WORD CONFUSION NETWORKS

Keith Vertanen

University of Cambridge, Cavendish Laboratory
Madingley Road, Cambridge, CB3 0HE, UK
kv227@cam.ac.uk

ABSTRACT

A limitation of most speech recognizers is that they only recognize words from a fixed vocabulary. In this paper, we explore a technique for addressing this deficiency using automatically derived units made up of letters and phones. We show how these units can be used for letter-to-phone conversion and open-vocabulary recognition. We further show how these units can be merged to form novel words while maintaining a word lattice structure. This allows creation of a word confusion network containing both in- and out-of-vocabulary (OOV) words. Experiments show these open vocabulary confusion networks improve recognition accuracy. They also allow open vocabulary recognition to be used in concert with a convenient confusion network result representation.

***Index Terms*—** Speech recognition

1. INTRODUCTION

Even if a speech recognizer's vocabulary is large, users may still say words not in the recognizer's vocabulary: proper names, technical jargon, or newly coined words, for example. A recognizer with a closed-vocabulary will always get these OOV words wrong. Using a more flexible, open-vocabulary recognizer, it is possible to make language-grounded hypotheses about novel words. In this work, we build and extend such an open-vocabulary recognizer. Using the approach in [1], we first learn a set of pairings between the written, grapheme sequences of a language and its spoken, phoneme sequences. For example, in English some likely pairings are:

$$\begin{array}{l} \textit{Graphemes} \\ \texttt{Phonemes} \end{array} \begin{pmatrix} \textit{ing} \\ \texttt{ih ng} \end{pmatrix} \begin{pmatrix} \textit{ation} \\ \texttt{ey sh ah n} \end{pmatrix} \begin{pmatrix} \textit{sch} \\ \texttt{sh} \end{pmatrix}$$

As in [1], we refer to these pairings as graphones. Graphones are learned from a pronunciation dictionary and added alongside the recognizer's normal vocabulary. A language model (LM) is trained on text where each OOV is replaced by its most likely graphone sequence. This "word+graphone" LM is used for recognition, returning both words and graphones. Graphones are then combined to form novel recognized words.

After showing our graphone-based open-vocabulary recognizer provides state-of-the-art performance, we extend this open-vocabulary technique by merging graphones into complete OOV words while maintaining a word lattice structure. A confusion network (CN) is then created containing both in- and out-of-vocabulary words. This CN not only further reduces word error rate (WER), but also provides a representation in which recognized words have both posterior probabilities and a set of competing word hypotheses. A possible application of such a representation might be a dictation interface which allowed open-vocabulary recognition and where in-vocabulary and out-of-vocabulary alternatives to the initially recognized words were presented using the CN.

In this paper, first we describe training of English graphones and give letter-to-phone conversion results. Second, we train word+graphone LMs and perform recognition experiments using these LMs. Third, we give an algorithm that merges graphones into words while maintaining a word lattice structure. Finally, we show confusion networks created from the merged lattices reduce open-vocabulary WER.

2. GRAPHONES

Before building the word+graphone LM used during recognition, a set of graphones and their probabilities is inferred using the joint multigram model [2]. A graphone inventory $\{G\}$, is a set of units g_i, each a pairing of letters/phones:

$$g_i = \begin{pmatrix} \ell_1, \ell_2, ..., \ell_j \\ \rho_1, \rho_2, ..., \rho_k \end{pmatrix}$$

Graphones have some minimum and maximum number of letters and phones. We used the same range for both letters and phones and denote a model by its range (e.g. 0-2). We found the inventory $\{G\}$ by generating all ways every dictionary word can be split into letter/phone chunks subject to the length restrictions. If $C(g_i)$ is the count of times g_i appeared in all segmentations, an initial unigram estimate is:

$$P(g_i) = \frac{C(g_i)}{\sum_{g_j \in \{G\}} C(g_j)} \qquad (1)$$

Graphones in a sequence S of length m are assumed to be independent of each other:

$$P(S = g_1, g_2, ..., g_m | m) = \prod_{i=1}^{m} P(g_i) \qquad (2)$$

	0-1	0-2	0-3	1-2	1-3
2-gram	18.76	10.19	8.70	10.84	9.83
3-gram	10.22	7.67	8.40	9.03	9.59
4-gram	7.61	7.55	8.40	8.97	9.58
5-gram	7.00	7.55	8.41	8.95	9.55
6-gram	6.86	7.54	8.39	8.96	9.55
7-gram	6.83	7.55	8.40	8.96	9.55

Table 1. PER of letter-to-phone conversion on CMU dev set, varying graphone size (columns) and LM size (rows).

Smoothing method	non-interpolated PER ± 1 sd	interpolated PER ± 1 sd
Witten–Bell	8.00 ± 0.13	7.68 ± 0.12
absolute discounting	11.71 ± 0.14	7.52 ± 0.12
Kneser–Ney	8.62 ± 0.13	6.84 ± 0.12

Table 2. Effect of smoothing and interpolation (6-gram 0-1).

The likelihood of word W with letters L and phones P is the sum over all sequences $\{S\}$ where $\{S\}$ contains all graphone sequences which exactly produce L and P:

$$\mathscr{L}(W) = \sum_{S\in\{S\}} P(S) \tag{3}$$

The most likely sequence S^* for W is:

$$S^* = \mathrm{argmax}_{S\in\{S\}} P(S) \tag{4}$$

In Viterbi-style training, S^* is found for each word, creating new counts which are used to reestimate the unigram probabilities. Graphones are pruned if their count falls below some threshold. Instead of making a hard decision about the segmentation of each training example, a soft decision can be made instead. This yields an expectation maximization (EM) formulation, see [2] for details.

To model dependency between graphones, the trained graphones are used to create a n-gram LM where the language consists of a "word" for each graphone. The most likely graphone sequence is found for each dictionary word using (4) and used as training text for a graphone LM. Using this LM, a new segmentation of the dictionary is found, a new LM is trained, and the process repeated until convergence. As suggested by [3], the LM-span is iteratively increased, first training a 2-gram to convergence, then a 3-gram, and so on.

Using the graphone LM, a word's most likely phones can be found from its letters (and vice-versa). This is done by finding all graphones sequences consistent with the known symbols and choosing the most likely sequence given the LM. A special end-of-word (EOW) letter can be used to help segment streams of graphones into complete OOV words. This is done by simply adding the EOW letter (denoted by •) to the end of every dictionary word and training as usual.

Iter. LM	LM cutoffs	EOW	Training	PER ± 1 sd
yes	0	yes	EM	6.84 ± 0.12
no	0	yes	EM	6.86 ± 0.12
yes	1/1/1/2/2/2	yes	EM	7.30 ± 0.12
yes	0	no	EM	6.66 ± 0.12
yes	0	yes	Viterbi	6.85 ± 0.12

Table 3. Effect of iterative LM training, LM cutoffs, end-of-word letter, and graphone training method (6-gram 0-1).

2.1. Letter-to-phone experiments

To recognize OOV words, our model needs to make accurate phonetic transcriptions of OOV words. So we first describe the ways in which we achieved good letter-to-phone conversion using graphones. Our experiments used the CMU dictionary, removing words with letters other than A–Z and apostrophe. We retained multiple pronunciations per word. We randomly split into a 80% training set, 10% development test set, and 10% evaluation test set. Evaluation used the phone error rate (PER), scoring a word's PER as the minimum of any of its pronunciation variants. Throughout, we report one-sigma confidence intervals using the bootstrap method [4].

We tested a variety of graphone and LM sizes (table 1). 0 letters/phone models outperformed those with a 1 letter/phone minimum. As in [1, 3], small graphones with long-span LMs proved best. Gains were small beyond 6-grams, so we chose a 6-gram 0-1 model for letter-to-phone conversion. Testing different LM smoothing methods (table 2), we found interpolation helped and (original) Kneser–Ney performed the best.

We found iterative LM training did not substantially reduce PER (table 3). Using no LM count cutoffs was helpful. Using the EOW letter hurt performance, so we will add EOW only when we must find word boundaries. Training graphones with EM was only slightly better than Viterbi. On the unseen eval set, we had a PER of 6.64% ± 0.08%. This is similar to other CMU letter-to-phone results (5.9% [3] and 7.0% [5]).

3. WORD+GRAPHONE RECOGNITION

We now create a word+graphone LM (overview in figure 1). We did this slightly different than previous work (e.g. [1]), using a two-step process using two different graphone models. First, given a corpus of LM training text and an in-vocab word list, phones are found for each OOV. If available, an OOV's phones are taken from CMU. Otherwise, phones are found with our best letter-to-phone model (6-gram 0-1, no EOW).

Next, the training text's OOVs are replaced by their most likely graphones subject to the previously found phones. This replacement uses a model different from the one used for letter-to-phone conversion. As we'll see, different sized graphones provide different WER. Finally, a word+graphone LM is trained using the text with OOVs replaced by graphones.

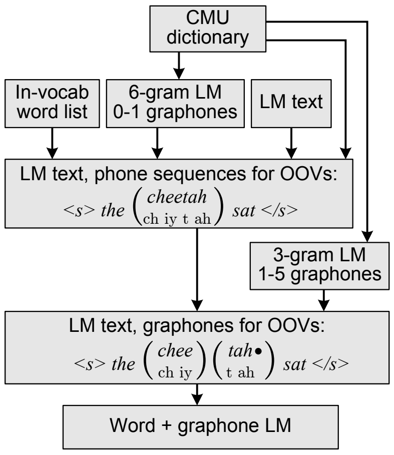

CMU dictionary
In-vocab word list
6-gram LM 0-1 graphones
LM text
LM text, phone sequences for OOVs:
<s> the (cheetah / ch iy t ah) sat </s>
3-gram LM 1-5 graphones
LM text, graphones for OOVs:
<s> the (chee / ch iy)(tah• / t ah) sat </s>
Word + graphone LM

Fig. 1. Overview of the training of a word+graphone LM.

Model	Graphones	Best path WER ± 1 sd	Conf net WER ± 1 sd
20K baseline	-	11.22 ± 0.40	11.01 ± 0.39
20K + 1-3 graphone	9K	9.77 ± 0.36	9.47 ± 0.35
20K + 1-4 graphone	19K	9.19 ± 0.34	9.01 ± 0.34
20K + 1-5 graphone	28K	8.97 ± 0.34	8.79 ± 0.33
20K + 1-6 graphone	31K	8.88 ± 0.33	8.70 ± 0.33
64K baseline	-	8.64 ± 0.33	8.30 ± 0.32
64K + 1-3 graphone	9K	8.74 ± 0.33	8.57 ± 0.33
64K + 1-4 graphone	19K	8.59 ± 0.32	8.50 ± 0.32
64K + 1-5 graphone	26K	8.55 ± 0.33	8.40 ± 0.32
64K + 1-6 graphone	29K	8.49 ± 0.33	8.35 ± 0.32

Table 4. WER of baseline and word+graphone models in best path and confusion network experiments on `si_dt_20`.

3.1. Best path recognition experiments

For recognition, we used HTK v3.4, HDecode, and the acoustic model recipe from [6]. We trained cross-word triphones on WSJ/TIMIT (218 hours), using 12 MFCCs plus deltas and delta-deltas, 32 Gaussians/state, and 10K tied-states. Decoding was in less than 6xRT on a 3GHz machine. We trained baseline and word+graphone LMs on CSR-III (222M words), using the top 20K/64K vocab and interpolated modified Kneser-Ney with cutoffs of 1/1/3. We used a 2-gram for decoding, rescoring lattices with a 3-gram. We report results on the combined WSJ0 and WSJ1 20K-vocab dev sets (`si_dt_20`, 894 sentences, 2.3% OOV at 20K vocab, 0.3% at 64K).

We used a word+graphone LM for recognition, producing a word lattice. We found the best lattice path, yielding a sequence of words and graphones. We converted graphones to their constituent letters, merging adjacent graphones and separating into complete OOV words using the EOW letter.

Using 20K in-vocab words, the word+graphone LMs reduced WER by 21% relative (best path column, table 4). This compares favorabley with the 15% reduction reported in [1] on a similar task (however we used 31K graphones compared to 12K in [1]). Using LMs with longer graphones reduced WER. Using 64K in-vocab words, gains were small, probably due to the low 0.3% OOV rate. While we thought our new two-step process of finding the best phones then aligning final graphones would be better, a test of directly converting OOVs with a 1-5 model yielded the same WER.

4. OPEN VOCABULARY CONFUSION NETWORKS

A confusion network (CN) is a compact representation of recognition results in which competing word hypotheses and their posterior probabilities appear in time-ordered sets [7]. CNs are built using words' time/phonetic overlap in a recognizer's lattice output. The CN's best or *consensus hypothesis*, is the highest probability hop in each set.

To build a CN containing OOVs, the graphones in a recognition lattice (figure 2a) must be merged into full words. In an iterative process, a graphone node without the EOW letter is selected. For every graphone following this node, a new node is created, concatenating the two nodes' letters and phones. Edges are created to/from the new node, maintaining the original lattice paths and scores (see algorithm 1 for details). The next node without EOW is selected, and the process continues. Upon completion, the lattice contains full OOV words with known pronunciations (figure 2b). An open vocabulary confusion network (OVCN) is created using a standard algorithm [7] which builds a CN from the lattice (figure 2c).

Algorithm 1 can be costly as it creates a node for every path between the chosen starting graphone and all reachable ending graphones. An approximate search is used in which a partially merged OOV word is pruned if its average per letter combined acoustic and LM score becomes too unlikely compared to a successfully merged OOV. The pruning is applied

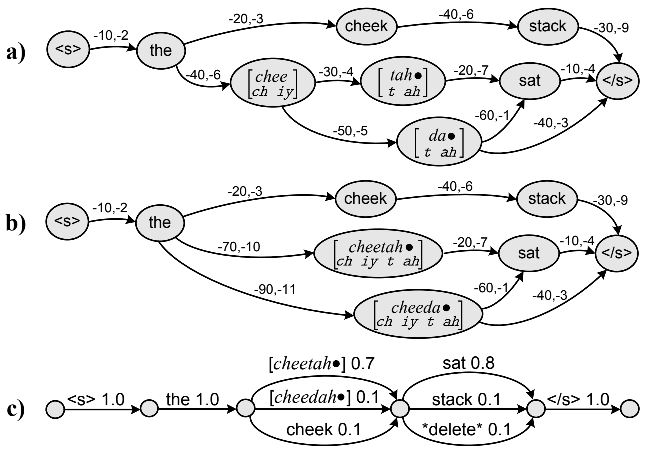


Fig. 2. a) Lattice before OOV merging, b) lattice after merging, c) open vocabulary confusion network.

based on the best, completely merged OOV whose start time is close to the partial OOV's start time. This helps insure good OOVs are obtained for the different lattice time regions.

4.1. Confusion network recognition experiments

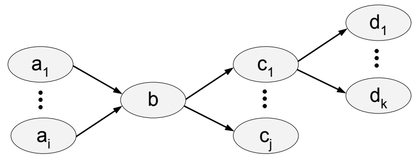
We found the consensus hypotheses of OVCNs made small, but consistent WER gains over best-path results of a word+graphone LM (table 4). Compared with the baseline CN, the 20K OVCN improved WER up to 21% relative. However, the 64K OVCN never improved upon the baseline CN.

To investigate performance at higher OOV rates, we used utterances collected in prior work [8]. This consisted of novices reading WSJ Spoke 2 sentences (`Eval`, 2016 sentences, 6.0% OOV at 20K, 1.9% at 64K). On this harder test set, OVCNs continued to provide small gains over using a best-path result (table 5). Compared to the baseline CN, WER was reduced by 27% relative at 20K and 4% at 64K.

Looking at the errors types in table 5, about half the gain of word+graphone recognition was by reducing insertions. An informal comparison showed that OOV models often consolidated difficult utterance parts into one-word OOVs rather than multiple, short and possibly erroneous in-vocab words.

5. CONCLUSIONS

We described a graphone model consisting of chunks of letters and phones. We used this model to train a word+graphone LM which could recognize both in- and out-of-vocab words. We gave an algorithm which merged graphones into OOV words while maintaining a lattice structure. This allowed creation of confusion networks containing OOV words. These open vocabulary confusion networks provided substantial accuracy improvements over baseline fixed-vocab recognition. They provided small, but consistent improvements over the best-path result of a word+graphone model. They also allow open vocab recognition to be used in concert with a convenient confusion network result representation.



```
N = lattice nodes, words/graphones ending at a time
E = lattice edges, with acoustic and LM scores
B = all nodes that are graphones without EOW letter
while B not empty do
    b = some node from B
    for each child c of b do
        Create new node n where n:
          Ends at time of node c
          Concatenation of letters/phones in b and c
        for each child d of c do
            Create edge n→d with same score as c→d
        for each ancestor a of b do
            Create edge from a→n with sum of scores
            of edges a→b and b→c
        if n does not have end letter then
            Add n to B
    Remove all edges to and from b, delete b
Remove all nodes no longer on any path through lattice
```

Algorithm 1: Merging graphones in a word lattice.

Model	Del	Sub	Ins	WER ± 1 sd
20K base, best path	1.1	14.4	5.0	20.49 ± 0.41
20K base, CN	0.9	14.3	5.1	20.32 ± 0.41
20K + 1-5, best path	1.3	11.5	2.4	15.19 ± 0.35
20K + 1-5, OVCN	1.2	11.2	2.4	14.83 ± 0.34
64K base, best path	1.1	11.7	2.9	15.69 ± 0.35
64K base, CN	1.0	11.5	2.9	15.39 ± 0.34
64K + 1-5, best path	1.2	11.2	2.6	14.96 ± 0.34
64K + 1-5, OVCN	1.1	11.1	2.5	14.72 ± 0.33

Table 5. Breakdown of types of errors made on `Eval`.

6. REFERENCES

[1] M. Bisani and H. Ney, "Open vocabulary speech recognition with flat hybrid models," *Proc. of Eurospeech*, pp. 725–728, Sep. 2005.

[2] S. Deligne and F. Bimbot, "Inference of variable-length linguistic and acoustic units by multigrams," *Speech Communication*, vol. 23, no. 3, pp. 223–241, 1997.

[3] S.F. Chen, "Conditional and Joint Models for Grapheme-to-Phoneme Conversion," *Proc. of Eurospeech*, pp. 2033–2036, Sep. 2003.

[4] M. Bisani and H. Ney, "Bootstrap estimates for confidence intervals in ASR performance evaluation," *Proc. of ICASSP*, vol. 1, pp. 409–411, May 2004.

[5] L. Galescu and J. Allen, "Bi-directional conversion between graphemes and phonemes using a joint n-gram model," *Proc. of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, 2001.

[6] K. Vertanen, "Baseline WSJ Acoustic Models for HTK and Sphinx: Training Recipes and Recognition Experiments," Tech. Rep., Cavendish Laboratory, 2006.

[7] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373–400, 2000.

[8] K. Vertanen, "Speech and speech recognition during dictation corrections," *Proc. of Interspeech*, Sep. 2006.