Subword-based Automatic Lexicon Learning for ASR

Timo Mertens [#,*1] and Stephanie Seneff [*2]

[#] *Norwegian University of Science and Technology*
Department of Electronics and Telecommunication
Trondheim, Norway
[1] tpm@mit.edu

[*] *Spoken Language Systems Group*
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
[2] seneff@csail.mit.edu

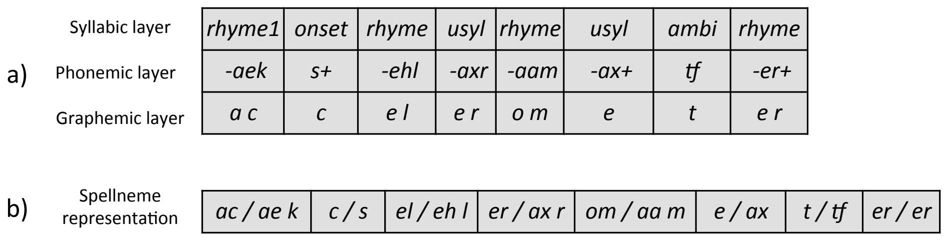
***Abstract*—We present a framework for learning a pronunciation lexicon for an Automatic Speech Recognition (ASR) system from multiple utterances of the same training words, where the lexical identities of the words are unknown. Instead of only trying to learn pronunciations for known words we go one step further and try to learn both spelling and pronunciation in a joint optimization. Decoding based on linguistically motivated hybrid subword units generates the joint lexical search space, which is reduced to the most appropriate lexical entries based on a set of simple pruning techniques. A cascade of letter and acoustic pruning, followed by re-scoring N-best hypotheses with discriminative decoder statistics resulted optimal lexical entries in terms of both spelling and pronunciation. Evaluating the framework on English isolated word recognition, we achieve reductions of 7.7% absolute on word error rate and 14.4% absolute on character error rate.**

I. Introduction

The word-level lexicon of an LVCSR system models the set of words to be recognized along with their pronunciations. If a word is not in the lexicon, the recognizer will make a substitution, insertion or deletion error. This is known as the out-of-vocabulary (OOV) problem. Numerous works have addressed this aspect of LVCSR in various application scenarios. One scenario is to detect which parts of the utterance correspond to words unknown to the recognizer [1], [2], [3]where popular approaches include using generalized word or subword filler models, aligning word and subword representations or using confidence measures obtained from various sources. Another scenario is the derivation of a lexical representation for the speech component of an unknown word. This, in turn, consists of two parts: first, a pronunciation is hypothesized using phonemic subword units, and second, said pronunciation is converted to a spelling. Only generating a pronunciation for an unknown word is sufficient in applications such as Spoken Term Detection (STD) [6], where phonemic representations of speech are adequate for indexing and search. For transcription, however, an orthography needs to be estimated from a given phonemic subword sequence, as in [7], where phone transcriptions are converted to spellings using memory-based learning for Dutch OOV words. Another approach is to take the subword decoder output and perform a dynamic alignment between a lattice and the words of a large fallback lexicon [8]. Lately, so-called flat hybrid models, originally used for letter-to-sound (L2S) conversion, have gained popularity for detecting and transcribing OOVs [9]. Such models essentially represent subword units that encode both pronunciation and spelling. The idea is to include such units in the word-level recognition lexicon and language model (LM) and let the decoder decide when to output a word or a subword sequence. The subword-spellings of such subword sequences can then be merged into lexical representations for the OOV.

The goal in this contribution is to learn a pronunciation lexicon from some speech examples. Instead of only focusing on either the spelling or the pronunciation aspects of lexical entries, we propose techniques that optimize both jointly. The motivation behind this is that we want to move away from a static recognition lexicon and explore automatic approaches to discovering an optimal lexical representation for a data set. In scenarios like spoken term discovery [10], where acoustic examples of the same word or phrase are clustered together, automatically proposing both spellings and pronunciations could be used as a first step to learning lexical entries. Furthermore, in applications like STD where a database of audio documents needs to be indexed, the OOV problem could be overcome by learning the words contained in the database automatically during indexing. In the Spoken Language Systems group at MIT numerous lines of research have investigated learning certain aspects of the lexicon. [11] learns pronunciation baseforms and variants using statistically learned subword units, [12] uses linguistically motivated hybrid subword units to propose spellings for perfect phone transcripts and [13] learns both spellings and pronunciations for new words using the same linguistic subword units as well as a statistical letter model.

Building on this existing body of research, we propose to learn a lexical inventory from scratch. We present a framework that decodes training speech utterances with linguistically motivated hybrid subword units, and utilizes pruning approaches to reduce this lexical search space. We investigate the usefulness of learning pronunciations and spellings both isolated and jointly from training data that only contains clusters of utterances, but no lexical information. Although we address only isolated word learning, the ultimate vision is a system that



a)

Syllabic layer	*rhyme1*	*onset*	*rhyme*	*usyl*	*rhyme*	*usyl*	*ambi*	*rhyme*
Phonemic layer	*-aek*	*s+*	*-ehl*	*-axr*	*-aam*	*-ax+*	*tʃ*	*-er+*
Graphemic layer	*a c*	*c*	*e l*	*e r*	*o m*	*e*	*t*	*e r*

b)

Spellneme representation	*ac / ae k*	*c / s*	*el / eh l*	*er / ax r*	*om / aa m*	*e / ax*	*t / tʃ*	*er / er*

Fig. 1. Subword Parse Table for 'communication'.

is capable of learning lexical entries from continuous speech, making the OOV problem obsolete for certain applications such as STD.
In the remainder of this paper we first present the subword units used to drive the ASR system. Sec. 3 describes the framework used to learn a lexicon from a speech database, followed by the experimental setup. Sec. 5 explains the experiments and presents evaluation results. The paper is concluded in Sec. 6.

II. Subword-based ASR for Lexicon Learning

The goal of this contribution is to investigate techniques for automatic lexicon learning: given some speech data, generate word-level lexical entries that best represent the data. To create the lexical search space, we use subword-based ASR to first decode speech into sequences of subword units and then use statistics from the decoding output to select the most adequate lexical representations for each word. This section describes the subword units used in this work.
As noted in Sec. 1, a number of subword units have been proposed for various ASR-related tasks. Of special interest for recognizing OOVs are hybrid units that encode both spelling and pronunciation. Although the focus has been largely on learning such a unit inventory automatically from some exemplar spellings/pronunciations, other, more linguistically motivated subword segmentations have been investigated as well [12], [14]. In this contribution we use so-called *spellnemes*, proposed in [12]. The underlying idea behind spellnemes is to segment a word into hybrid units conditioned on a set of linguistic constraints, where each unit is associated with both spelling and pronunciation. These constraints are enforced through a linguistic model that essentially represents a parse table for a given word. A parse table consists of various layers which model different subword aspects such as syllabification, subsyllabic position, stress and morphology. An example parse for the English word *accelerometer* can be found in Fig. 1 a).

As can be seen in Fig. 1 b), the lower two layers of the parse table can be used to segment a given word into both spelling and pronunciation chunks, where each cell contains a so-called spellneme. The inventory of spellnemes is obtained from a large language-specific pronunciation lexicon. Each word is parsed using a context-free grammar which was developed manually. The grammar basically describes permissible letter/phone alignments. If a word fails to parse according to the grammar, the corresponding rules are updated manually. This procedure is applied iteratively across the whole lexicon. Each word can be represented as a sequence of spellnemes as long as the spelling or pronunciation can be parsed with the grammar. This serves de-facto as a reversible L2S or S2L module.

In comparison to graphones [15], another hybrid unit, spellneme segmentations are constrained by linguistic features of the language in question. That is, where graphones are learned statistically according to some optimization criterion, the set of possible spellnemes is based predominantly on the syllable structure of the language. An important question in subword-based ASR is how much context a subword unit should model. At one end of the spectrum, phone units model the least amount of acoustic context but allow for the highest degree of generalization. On the other side, larger units like morphemes [14] or spellnemes increase lexical constraints for the decoder, but struggle with unseen events. Graphones usually fall in between, since letter/phone alignment lengths, and thus the modeling context, are usually chosen depending on the task. Once an inventory of spellnemes has been derived from some training data, a spellneme language model needs to be trained. The same data that was used for generating the unit inventory can be utilized to train an N-gram model. In this contribution we only consider isolated word recognition, which means that we do not need to model cross-word effects and can therefore straightforwardly train the model on the lexicon. As explained before, we want to learn a task-specific recognition lexicon from scratch based on subword ASR. By using an N-gram model over spellnemes, one could argue that word-like units are encoded in the LM with a strong bias towards observed words. Although this is correct, the AMs of the recognizer strongly influence the final decoder output, which means that unseen spellneme sequences are likely to be output for words that were not observed during spellneme training. Our idea of lexicon learning, ultimately, is that we train a subword model on a big static training lexicon, potentially containing millions of words, and use that general model to obtain the most precise and concise lexical representation for a given data set. The model will perform better on frequent words, but, at the same time, should also be able to generate sensible representations for infrequent words.

III. Lexicon Learning

Fig. 2 illustrates the proposed lexicon learning framework. Given a database of clustered but unknown isolated word utterances, we use the spellneme decoder described in Sec. 2 to generate an N-best list for each utterance. The core of the framework is the so-called *hypothesis pruning* stage where the goal is to filter the hypotheses proposed by the ASR decoder according to some statistics obtained across the decoder output

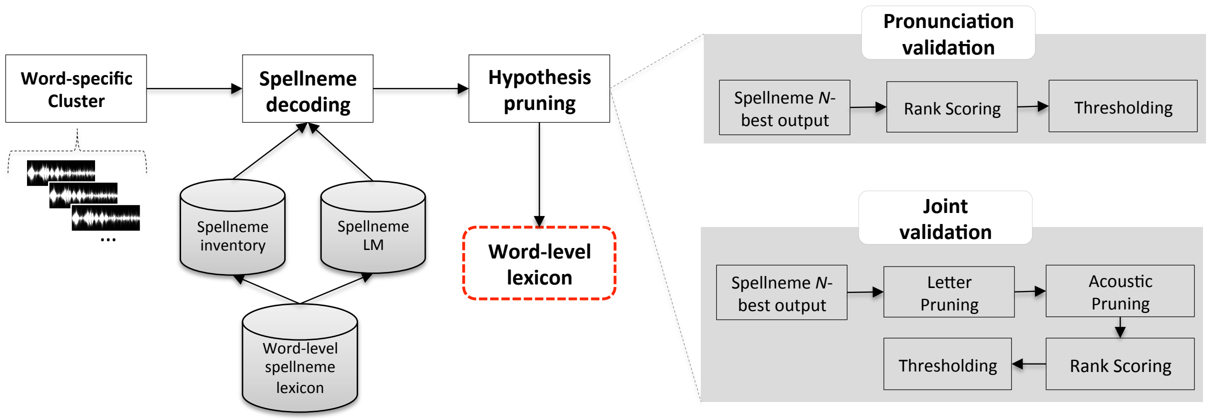


Fig. 2. Overview of the lexicon learning framework. A spellneme decoder produces subword hypotheses for both spelling and pronunciation. Two pruning methods are investigated: pronunciation validation and joint validation.

of either the utterances within the same cluster or across the utterances of all clusters. The output of this filter is a word-level lexicon which, ideally, encodes an exhaustive set of lexical entries tailored to the data in question. The rest of this section describes two approaches to hypothesis pruning: first, we focus on learning the pronunciation for a set of words. Then, we increase the complexity of the problem by trying to optimize with respect to both spelling *and* pronunciation.

A. Pronunciation Validation

As the lexicon maps orthographies to pronunciations, it is important to model an optimal pronunciation space for each word. This problem is known as pronunciation modeling: if too many pronunciation variants per word are included, acoustic confusability at runtime leads to decreased accuracy. Including only canonical pronunciations, on the other hand, might not capture all variations robustly. The assumption when trying to predict the optimal set of pronunciations for a word is that the identity, i.e., the spelling, of the word is known (which is used to drive a letter-to-sound system followed by acoustic pruning). In our scenario, however, the utterances are clustered into sets where it is known that all utterances in a given set represent the same word, but the identity of this word is unknown. For this reason, we cannot constrain the search space of the subword decoder with the spelling, and instead have to use the N-gram spellneme LM shown in Fig. 2. The idea is then to constrain the pronunciation search space to those hypotheses that represent the most reliable pronunciations for the cluster.

The core of the pronunciation validation approach is a scoring method based on discriminating between the average ranks of correct and incorrect pronunciation hypotheses in N-best lists. Note that a pronunciation can be obtained for a sequence of spellnemes by simply concatenating the pronunciations of each individual subword. *Rank scoring* is formulated as follows. Let S_c be the set of spellneme N-best hypotheses for word cluster c, where $s_c \in S_c$ and $h \in s_c$ denote a specific N-best list and a specific pronunciation hypothesis in s_c, respectively. The fitness of a hypothesis h is then defined as

$$\text{in\_score}(h) = \frac{\text{count}(h \in S_c)}{\frac{1}{|S_c|}\sum_{s_c \in S_c} \text{rank}(h \in S_c)} \tag{1}$$

where the numerator reflects how often a hypothesis occurred in all relevant N-best lists, which is then normalized by the average rank (i.e., N in the N-best list) that hypothesis achieved in said lists. If a hypothesis is not in the N-best list we assign a rank of $n+1$. The intuition behind the in_score is that a hypothesis that is decoded often for the target cluster at a high position is probably more relevant than a hypothesis that occurs only rarely and at lower N. In the same spirit we also want to penalize hypotheses that occur often in irrelevant N-best lists, that is, $h_n \in S_{\bar{c}}$:

$$\text{out\_score}(h) = \frac{\text{count}(h \in S_{\bar{c}})}{\frac{1}{|S_{\bar{c}}|}\sum_{s_{\bar{c}} \in S_{\bar{c}}} \text{rank}(h \in S_{\bar{c}})} \tag{2}$$

The overall score is then

$$\text{score}(h) = \text{in\_score}(h) - \text{out\_score}(h), \tag{3}$$

which means that, if a pronunciation achieves a high score, it should be an adequate representation for the lexical entry. This score, however, is not a normalized probability, which makes it difficult to threshold. An intuitive way of choosing the set of pronunciations for a lexical entry is to sort all proposed pronunciations according to the above score and pick the top ranking ones. To avoid hardcoding the number of pronunciations, we decided to consider the distribution of scores across all proposed pronunciations for c. We first determine the standard deviation σ across the scores, and pick only those pronunciations whose scores are more than x multiples of σ away from the mean μ:

$$\text{decision} = \begin{cases} \text{accept} & \text{if score}(h) \text{ - } \mu \geq x \cdot \sigma \\ \text{reject} & \text{otherwise} \end{cases}$$

As irrelevant hypotheses will have scores in similar regions this technique only retains a pronunciation that has a large enough margin between its score and the mean. Using other decode information such as AM scores is left for future work.

Pronunciation validation optimizes a lexicon with respect to the suitability of a set of pronunciations for a given word across *all* clusters. Lexical entries, however, also require an orthographic representation. Hence, the next section describes how we extend the above methods to optimize both spelling and pronunciation jointly when learning lexical entries.

B. Joint Validation

As spellnemes represent hybrid units, the output of the spellneme decoder in Fig. 2 can be interpreted in terms of both spelling and pronunciation. Instead of only focusing on the pronunciation, as in the previous section, we also want to obtain a set of spellings that describe a cluster.

Letter pruning: To obtain a spelling for a decoder hypothesis h the spelling components of each spellneme in h can be merged (since we only discuss isolated word recognition we can ignore cross-word effects). As with pronunciation validation, we start with the set of N-best outputs for a given cluster S_c. The proposed spellings at each N of all $s_c \in S_c$ are used to train an N-gram letter model. By considering co-occurrence statistics over the proposed letter sequences for c we can prune those letter hypotheses that fall below a threshold, essentially removing spurious spellings. We use either the L2S module described in Sec. 2, or directly the pronunciations proposed by the decoder to synthesize pronunciations for the spellings. The reason for exploring two methods of selecting pronunciations is to understand whether the pronunciation search space should be restricted by means of using the spellneme output, or whether it should be opened up by synthesizing alternative pronunciations with the L2S module.

Acoustic pruning: The hypotheses are then used as candidates for an acoustic pruning stage, which is simply decoding the utterances belonging to c based on the proposed lexical entries that survived letter pruning. Letting the AMs decide which pronunciations fit c best moves from orthographic constraints enforced through letter-pruning to optimizing the pronunciation aspect of the lexical hypotheses. As opposed to rank scoring, both letter and acoustic pruning optimize the lexical representation for every c in isolation.

Rank scoring: Rank scoring as proposed in Eq. 3 can again be used to discriminatively filter the surviving hypotheses from the acoustic pruning stage. Now, the optimization is carried out across the whole lexicon, that is, over all c. This entails that we compile the survivors of acoustic pruning into a general word-level lexicon. Although we do not know the spelling representation of c, we can still use the discriminative nature of rank scoring by replacing the spelling of a lexical hypothesis with an id referencing the cluster it originated from. Instead of spellings, the intermediate lexicon therefore maps cluster-ids to pronunciations, where the ids for multiple pronunciations for a cluster are enumerated uniquely. We then decode the training data again with this lexicon and apply rank scoring to these N-best output lists, which is possible since we have a notion whether a hypothesis was decoded for the correct cluster or not.

To summarize, the proposed joint optimization first focuses on learning a spelling model for every cluster, which is used to prune unlikely entries. Acoustic pruning ranks the remaining hypotheses for a single cluster, and rank scoring prunes this set discriminatively according to decoder statistics across all clusters.

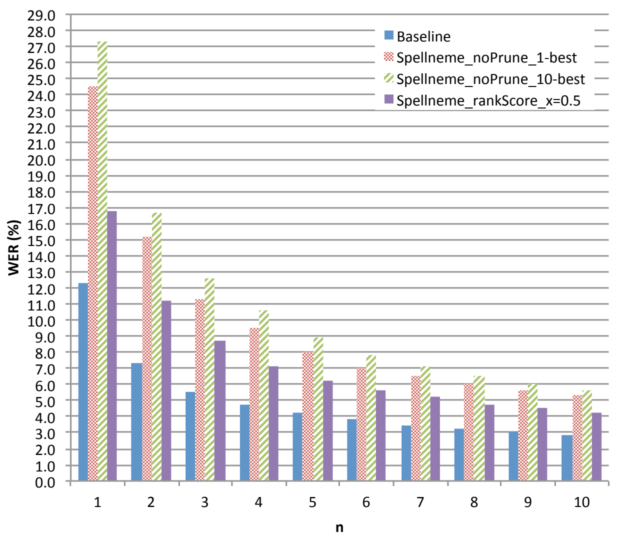


Fig. 3. WER (%) as a function of varying N-best list depth. `noPrune` uses only the output of the spellneme decoder by picking the m-best hypotheses as pronunciations. `rankScore` prunes the spellneme hypotheses and retains a hypothesis if score$(h) \geq x \cdot \sigma$

IV. Experimental Setup

We evaluate the proposed methods on the Phonebook corpus [16] which contains telephone-quality words spoken in isolation. We divided the corpus into a training portion, consisting of 2k unique words with 10 occurrences of each word, and a test portion with the same inventory of words but only two utterances per word.

The spellneme subword inventory is derived from the 300k most frequent words of the Google N-gram corpus, resulting in 2745 unique spellnemes. We trained a 3-gram spellneme LM based on the same data. An expert-based word-level pronunciation lexicon containing the in-corpus words only was used to measure baseline recognition performance. The SUMMIT segment-based decoder [17] was used in this work. We use context-dependent diphone AMs based on 14 Mel-Frequency Cepstral Coefficients averaged over 8 regions at hypothesized phonetic boundaries. Diagonal Gaussian mixture models with up to 75 mixtures per model were trained on telephone speech.

We use error rates to evaluate certain aspects of lexicon learning. Word error rate (WER) is used to measure the discriminability between learned pronunciations in a basic isolated word recognition scenario. Character error rate (CER) is used when assessing the quality of the proposed spellings, which is a more discerning measure for spelling learning, as a slight deviation from the true spelling gets a much higher score than a gross misspelling. When evaluating N-best lists, the minimum CER across all N is reported.

V. Experiments & Results

A. Pronunciation Validation

First, we want to evaluate the ability of a spellneme-based ASR system to learn the pronunciation-side of the lexical entries across all c without having any knowledge of the word's actual spelling. Each c is decoded with the spellneme system and N-best lists are produced, where $n = 10$ for all experiments. To prepare the baseform lexicon, the spelling of a proposed lexical entry at each N in each c was replaced by c's true word label and enumerated according to N. The resulting word-level lexicon was then used to decode the training data again. We applied rank scoring to the resulting N-best lists and only kept lexical entries whose scores were x standard deviations above the mean. We found $x = 0.5$ to be optimal. This lexicon is then used to decode the test data. In this experiment, WER measures the ability of the learned lexical inventory to discriminate between correct and incorrect pronunciations.

Fig. 3 displays WER as a function of the depth N in the N-best list. First, the expert baseline lexicon achieves the best results across all N. When simply taking the 1-best hypotheses of the spellneme output as lexical entries, a WER of 24.5% at $n = 1$ is achieved. This means that open spellneme decoding is able to produce useful pronunciations for the word clusters in question, however not of the same quality as expert pronunciations. Including all hypotheses from the spellneme output, denoted as `noPrune_10-best`, in the learned lexicon results in more confusable pronunciations and a subsequent increase by 2.8% absolute in WER compare to including only the best hypotheses. Rank scoring, on the other hand, is able to remove pronunciations that caused the decoder to confuse lexical entries, resulting in an absolute improvement of 7.7% for $n = 1$. Regardless of rank scoring, the spellneme-based method does not achieve the same performance as manually created pronunciations. Other automatic pronunciation learning approaches, e.g., [11], achieved better performance than the baseline lexicon. These approaches, however, condition the pronunciation search space on a the word's assumed known spelling, which is not the case in our experiment.

B. Joint Learning

As described above, we merge the spellings of all spellneme N-best lists for a given c and learn a letter N-gram model, where we chose $n = 3$. This model is used to retain only the 10 most likely spellings, which are then used to decode all utterances in c. Doing this over all c we obtain a general lexicon, which is then used to re-decode the training data, followed by rank scoring. We start by evaluating the performance of spelling learning, followed by an assessment of the pronunciations associated with the final lexicon.

Spelling learning: We are now interested in how well the framework learns the spelling component for c. We measure CER based on the output of the acoustic pruning stage, i.e., the decode of c in the training data. We deliberately ignore acoustic confusability and only evaluate the fitness of the spellings generated for that cluster. Results are illustrated in Fig. 4. Again, we report CER as a function of N. Applying no pruning yields a CER of 39.8% at $n = 1$, down to 26.9% at $n = 5$.

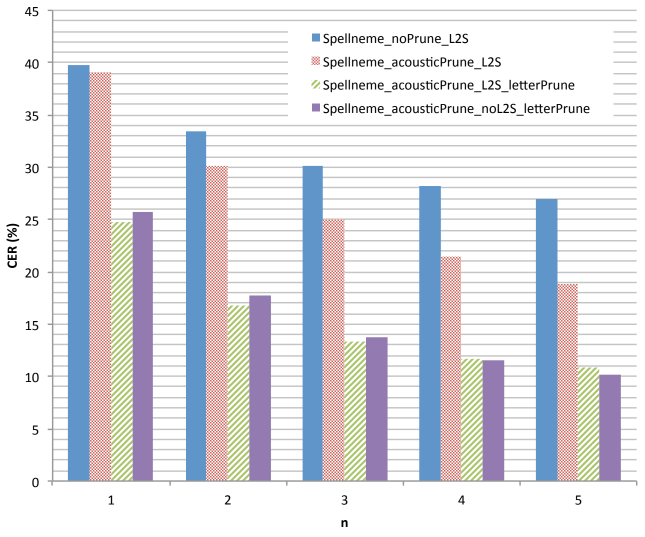


Fig. 4. Character Error Rate (CER; %) as a function of varying N-best list depth. `noPrune` evaluates the output of the spellneme decoder only. `acousticPrune` prunes the spellneme hypotheses by re-recognizing the training data. `letterPrune` scores the spellneme output with a 3-gram letter model and retains the top 10 hypotheses. `L2S` denotes using the L2S system to generate pronunciations for spellings.

TABLE I
WER (%) ON THREE SPELLING LEARNING SETUPS. ALL APPROACHES USE LETTER AND ACOUSTIC PRUNING.

Setup	WER	1-best CER
L2S	23.0	24.7
noL2S	20.8	25.7
noL2S + rank scoring	15.9	???

Using acoustic pruning does not affect the top-ranked results in the N-best lists significantly (39.1%), but has more effect at higher N: a decrease of CER of up to 8.1% absolute over no pruning is achieved at the highest N. This means that spurious spellings which were not high-scoring but still competitive could be removed with the help of the AM. Letter pruning, on the other hand, improves the 1-best output by up to 14.4% abs. to 24.7%, and improvements remain consistent for higher N. Constraining the input to the acoustic pruning stage by means of imposing tight letter pruning appears to work well, especially for finding the optimal 1-best hypothesis. Using the L2S system compared to simply picking the spellneme output to obtain the pronunciation appears to not affect CER significantly. In our framework, predicting the best letter representation can be ambiguous since a single hypothesized pronunciation can have multiple spellings. We chose to pick the spelling with the lowest letter LM score.

Pronunciation validation: The output of the spelling optimization is a lexicon based on a set of learned spellings and their corresponding pronunciations. We focus on the three setups that are of most interest: just using acoustic pruning, and additionally using letter pruning, with the L2S or the spellneme pronunciations. The last aspect to evaluate in the framework is the quality of the overall lexicon in terms of discrimination capabilities between lexical entries. As for pronunciation validation, we replace the spellings in the lexicon with placeholder cluster identities, re-recognize the training data, optionally apply rank scoring and decode the test data with the resulting lexicon. Results are shown in Table 1. First, the difference between L2S and spellneme pronunciations, which was negligible according to CER is more obvious when considering WER: the system with L2S achieves a WER of 23.0% compared to 21.8% for the spellneme-based system. An explanation for this is that the spellneme pronunciations are already optimized toward what the AM favor, whereas the set of pronunciations synthesized through L2S allows for noisy hypotheses which increase confusion during subsequent acoustic pruning. As the letter representations are essentially the same, that difference only becomes apparent when examining the quality of the pronunciations w.r.t. the data and the AMs. Applying rank scoring on that lexicon results in another 5% abs. decrease in WER. The resulting lexicon performs slightly better than the one optimized for pronunciation only, which means that the additional constraint enforced by letter pruning does not remove pronunciations that are preferred by the AMs. Jointly optimizing the spelling and pronunciation by means of spelling pruning and rank scoring yields therefore a lexicon that has both the lowest CER on the training data and WER on the test set.

VI. Conclusion

We presented a framework for learning a pronunciation lexicon from a set of isolated word utterances without knowing either pronunciation or spelling of the words in question. The joint spelling/pronunciation search space was generated by decoding the speech with linguistically motivated subword units and a weakly constrained language model. We presented various methods that optimize both pronunciation and spelling either in isolation or jointly. On an isolated word recognition task, we achieved 7.7% abs. WER reduction when optimizing the pronunciations across the whole lexicon. Optimizing the spelling of the lexical entries resulted in up to 14.4% abs. reduction in CER over the unconstrained baseline. The best lexicon according to both acoustic discriminability and spelling accuracy was obtained through a chain of spelling and acoustic pruning as well as using decoder statistics to re-score potential lexical hypotheses.

Learning a lexicon from some unannotated but clustered data is especially useful in scenarios where a dynamic lexicon is required, such as STD. The appealing aspect of our framework is that only relevant terms are learned directly from the data while relying on a subword inventory and language model that can be trained on potentially millions of words. To make our methods transferrable to more scenarios, future work will target learning a lexicon from continuous speech. Directly incorporating other ASR statistics, such as confidence scores, could be used to improve pruning performance. Furthermore, alternative hybrid subword representation have to be investigated as an alternative to linguistically motivated units.

VII. Acknowledgments

The work for this contribution was supported by the SMUDI project under the Research Council of Norway's VERDIKT program.

References

[1] T. Hazen and I. Bazzi, "A comparison and combination of methods for oov word detection and word confidence scoring," in *Proc. ICASSP*, vol. 1. IEEE, 2001, pp. 397–400.
[2] H. Lin, J. Bilmes, D. Vergyri, and K. Kirchhoff, "Oov detection by joint word/phone lattice alignment," in *Proc. ASRU*. IEEE, 2007, pp. 478–483.
[3] C. White, G. Zweig, L. Burget, P. Schwarz, and H. Hermansky, "Confidence estimation, oov detection and language id using phone-to-word transduction and phone-level alignments," in *Proc. ICASSP*. IEEE, 2008, pp. 4085–4088.
[4] T. Schaaf, "Detection of oov words using generalized word models and a semantic class language model," in *Proc. Eurospeech*. Citeseer, 2001, pp. 2581–2584.
[5] F. Stouten, D. Fohr, and I. Illina, "Detection of oov words by combining acoustic confidence measures with linguistic features," in *Proc. ASRU*. IEEE, 2009, pp. 371–375.
[6] T. Mertens, R. Wallace, and D. Schneider, "Cross-site combination and evaluation of spoken term detection systems," in *Proc. CBMI*. IEEE, 2011.
[7] B. Decadt, J. Duchateau, W. Daelemans, and P. Wambacq, "Phoneme-to-grapheme conversion for out-of-vocabulary words in large vocabulary speech recognition," in *Proc. ASRU*. IEEE, 2001, pp. 413–416.
[8] O. Scharenborg and S. Seneff, "Two-pass strategy for handling oovs in a large vocabulary recognition task," in *Proc. Eurospeech*, 2005.
[9] M. Bisani and H. Ney, "Open vocabulary speech recognition with flat hybrid models," in *Proc. Interspeech*. Citeseer, 2005, pp. 725–728.
[10] A. Park and J. Glass, "Unsupervised pattern discovery in speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, 2008.
[11] I. Badr, I. McGraw, and J. Glass, "Learning new word pronunciations from spoken examples," in *Proc. Interspeech*, 2010.
[12] S. Seneff, "Reversible sound-to-letter/letter-to-sound modeling based on syllable structure," in *Proc. HLT*. Association for Computational Linguistics, 2007, pp. 153–156.
[13] G. Choueiter, S. Seneff, and J. Glass, "New word acquisition using subword modeling," in *Proc. Interspeech*, vol. 2007. Citeseer, 2007.
[14] T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, and J. Pylkkönen, "Unlimited vocabulary speech recognition with morph language models applied to Finnish," *Computer Speech & Language*, vol. 20, no. 4, pp. 515–541, Oct. 2006.
[15] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.
[16] J. Pitrelli, C. Fong, S. Wong, J. Spitz, and H. Leung, "Phonebook: A phonetically-rich isolated-word telephone-speech database," in *Proc. ICASSP*, vol. 1. IEEE, 1995, pp. 101–104.
[17] J. Glass, "A probabilistic framework for segment-based speech recognition," *Computer Speech & Language*, vol. 17, no. 2-3, pp. 137–152, 2003.