LANGUAGE MODEL VERBALIZATION FOR AUTOMATIC SPEECH RECOGNITION

Haşim Sak, Françoise Beaufays, Kaisuke Nakajima, Cyril Allauzen

Google
{*hasim,fsb,kaisuke,allauzen*}@google.com

ABSTRACT

Transcribing speech in properly formatted written language presents some challenges for automatic speech recognition systems. The difficulty arises from the conversion ambiguity between verbal and written language in both directions. Non-lexical vocabulary items such as numeric entities, dates, times, abbreviations and acronyms are particularly ambiguous. This paper describes a finite-state transducer based approach that improves proper transcription of these entities. The approach involves training a language model in the written language domain, and integrating verbal expansions of vocabulary items as a finite-state model into the decoding graph construction. We build an inverted finite-state transducer to map written vocabulary items to alternate verbal expansions using rewrite rules. Then, this verbalizer transducer is composed with the n-gram language model to obtain a verbalized language model, whose input labels are in the verbal language domain while output labels are in the written language domain. We show that the proposed approach is very effective in improving the recognition accuracy of numeric entities.

Index Terms— language modeling, verbalization, finite-state transducer, speech recognition

1. INTRODUCTION

Automatic speech recognition (ASR) systems need to transcribe verbal language into properly formatted written language for presentation to the user. This is difficult. First, lexical and non-lexical vocabulary items need to be converted to their pronunciations for phonetic acoustic modeling. Second, the format of the raw output from the speech decoder depends on the choice of vocabulary units used for language modeling. Third, there is ambiguity in converting between verbal and written language in both directions.

Consider for example the following verbal transcript:

> *"ten wall street apartment forty five eighty one new york new york one hundred twenty five"*

Transcribing this utterance properly is an ambiguous task. The conversion of numeric entities to written form is especially non-deterministic. For instance, one can transcribe the numeric entities in verbal or written domain, e.g. *"one"* vs *"1"*. Also, the proper format depends on the application domain. For example, some words can be abbreviated or replaced by an acronym depending on the application, e.g. *"NY"*. Finally, the correct transcription can also depend on real world knowledge such as the existence of an address in the transcribed format. In this example, *"forty five eighty one"* can be realized as *"4581"* or *"45 81"*. A proper written transcription of this utterance for a voice-search-enabled maps application may be as follows:

> *"10 Wall St. Apt. 4581 New York, NY 10025"*

In an ASR system, the formatting of transcripts is generally determined by a combination of pre-processing of the language model training text, language modeling approach, and post-processing of the recognition transcripts. One can think of three main approaches to handle transcript formatting.

In the first approach, *verbal-domain language modeling*, all non-lexical vocabulary items are expanded to their verbal forms by pre-processing the language model training data using text normalization rules [1, 2]. Hence, the language model is trained on verbalized text similar to the above verbal transcript and the speech recognition transcripts are in verbal form as well. The recognition transcripts thus need to be converted to proper written language for presentation to the user. This requires using inverse text normalization rules and/or rescoring techniques, possibly with class-based language models as in [2]. This approach has some disadvantages. The pre-processing is an ambiguous text normalization process and it loses written language formatting information. The post-processing is also ambiguous and using rescoring techniques to disambiguate alternate transcripts in written language complicates the system, especially when combined with class-based models.

The second approach relies on *class-based language modeling* [3] to model non-lexical vocabulary items using regular languages to represent classes such as date, time, number [4]. There are also some potential drawbacks to this approach, such as its inability to retain contextual information in the language model for frequently occurring class instances. Besides, the static construction of the decoding network may not be feasible for a large number of classes due to memory limitations, and dynamic composition during decoding can be slow.

The third approach is *written-domain language modeling*. This approach aims at retaining the written language formatting information in the language model and leveraging the disambiguation power of the language model to properly format the recognition transcripts by handling the pronunciation of non-lexical vocabulary items directly in the pronunciation lexicon. This approach is attractive for extremely large language models (with vocabulary lists on the order of millions of words, and training corpora on the order of billion of words [1]), since large data sizes alleviate data sparsity and out-of-vocabulary (OOV) issues. However, it is not practical nor optimal to handle these entities in the lexicon due to the combination of verbalization alternates with pronunciation alternates, and the resulting complexity of the lexicon.

In this paper, we propose to train the language model in the written domain, and to create a finite-state verbalization model that we incorporate in the decoding graph by composition with the language model. The verbalization model effectively transforms written non-lexical items into verbal items that can be looked up in the lexicon. The approach thereby maintains the desired richness of a written language model, together with the simplicity of a verbal-domain lexicon.

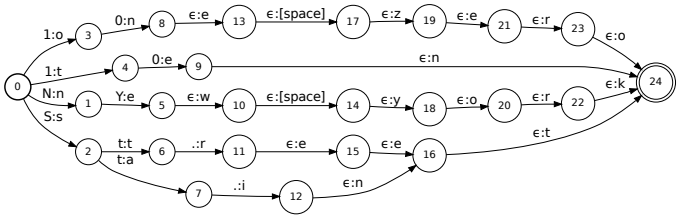

Fig. 1. Finite-state transducer for the sample rewrite grammar rule.

2. LANGUAGE MODEL VERBALIZATION

Weighted finite-state transducers (WFSTs) are widely used to represent all the components of a speech recognition system [5]. This representation comes together with a composition algorithm that efficiently combines all the components into a single finite-state decoding network. More specifically, in a finite-state transducer based ASR system, a decoding network D is typically constructed by composing the following finite-state models: a context-dependency model C mapping context-dependent phones to context-independent phones, a pronunciation lexicon L mapping context-independent phone sequences to words, and an n-gram language model G assigning probabilities to word sequences. In the WFST framework, this is expressed as $D = C \circ L \circ G$, where $\circ$ is the composition operator. In this setup, the speech decoder handles decoding over the Hidden Markov Model (HMM) states corresponding to context-dependent phones. An alternative proposed in [6] is to also incorporate a Hidden Markov Model H mapping HMM state sequences to context-dependent phones into the decoding network construction, which can be expressed as $D = H \circ C \circ L \circ G$. In this work, we chose the first setup. Our implementation relies on the OpenFst weighted finite-state transducer library [7].

2.1. Approach

In this paper, we propose to incorporate a verbal expansion of vocabulary items in the construction of the decoding network. To this end, we construct a finite-state verbalizer transducer V that maps verbal expansion of word sequences to vocabulary items. With this model, the decoding network can be expressed as $D = C \circ L \circ V \circ G$. We use grammars to expand non-lexical items such as numeric entities. Such grammars rely on regular expressions and context-dependent rewrite rules, and are commonly used for text processing and verbal expansion for text-to-speech and speech recognition. They can be efficiently compiled into finite-state transducers [8, 9].

A sample rewrite grammar rule is given below for a small set of vocabulary items. (The rewrite rules for numeric entities can be expressed efficiently in compact form due to the regular language nature of these entities, rather than explicitly listing them as in the example.)

rule = (“10” : “ten”) | (“10” : “one zero”) | (“NY” : “new york”) | (“St.” : “street”) | (“St.” : “saint”) ;

This sample rule can be efficiently compiled into a finite-state transducer representation as shown in Figure 1 using a rule compiler [9]. We defined multiple grammar rules to verbalize various classes of vocabulary items and all these rules were compiled into a set of finite-state transducers.

$\mathcal{V} \leftarrow$ vocabulary of language model
$R \leftarrow$ set of finite-state transducers for rewrite grammar rules
for all $v \in \mathcal{V}$ **do**
  $W \leftarrow \varnothing$
  for all $r \in R$ **do**
    $w \leftarrow$ *rewrite*(v, r)
    if w is not empty string **then**
      insert w into W
    end if
  end for
  if W is $\varnothing$ **then**
    $W \leftarrow v$
  end if
  for all $w \in W$ **do**
    add a path $w \rightarrow v$ to V_{FST}
  end for
end for
$V_{FST} \leftarrow$ *closure*(*determinize*(V_{FST}))

Fig. 2. The verbalizer FST construction algorithm.

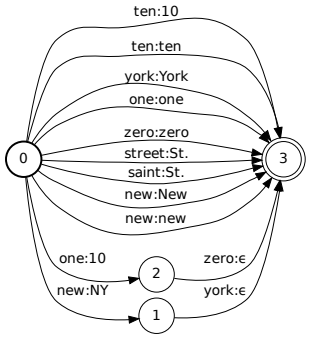

Fig. 3. Example finite-state transducer V for a simple non-determinized non-closed verbalizer model.

2.2. Verbalizer Model Construction

We use a simple algorithm to construct the finite-state transducer for the verbalizer as shown in Figure 2. The algorithm takes as input the vocabulary of the language model, the set of finite-state transducers for the compiled rewrite grammar rules. It enumerates all the vocabulary items and tries to rewrite each vocabulary item using each rewrite grammar rule transducer, and accumulates these verbal expansions in a set. If there is no rule that applies to a vocabulary item, the item itself is inserted in the set of verbal expansions. This is generally true for lexical vocabulary items, for which the lexicon can directly produce pronunciation alternatives. To construct the verbalizer transducer using all possible verbal expansions for each vocabulary item, we simply add paths from the start state to the final state of the transducer with each verbal expansion for a vocabulary item on the input label and that vocabulary item on the output label.

A simple finite-state transducer for the verbalizer model is shown in Figure 3. The input labels are in verbal domain and the

output labels are in the written domain of the language model. Please note that verbal expansions of vocabulary items are not context-dependent. We need a parallel corpus of sentences in written and verbal language to learn a context-dependent verbalization model. It also requires model smoothing to generalize verbalization over unseen instances especially for numeric entities. The verbalizer can also be used to normalize capitalization for the lexicon if capitalization is retained in the language model. Finally, it is also possible to assign weights to the rewrite rules and in the verbalizer model to discriminate between verbalization alternates. As seen in this example, the verbalizer model is non-deterministic in both input and output labels.

We use the generalized composition algorithm by Allauzen et al. [10] to compose the verbalizer transducer V with the language model G. This algorithm allows efficient composition by avoiding the creation of useless output-epsilon paths while also optimizing the composed model by pushing forward labels and weights for runtime efficiency. The resulting verbalized language model can be used with dynamic, runtime, composition in the decoder or it can be statically composed with the context-dependent lexicon to build a static decoding network.

3. EVALUATION

We applied the language model verbalization approach to the modeling of numerical entities in Google's French speech recognition system. This system used to rely on the written-domain language modeling approach described in Section 1. We chose this specific application after noticing that a large fraction of the French recognition traffic was related to Maps entities, and that the written-domain approach was causing a lot of formatting errors on such entries. Times, dates, and other numerical entities were also affected, and improved as a result of this work.

In the subsections below, we first describe the number error rate metric that we used to evaluate recognition accuracy for numeric entities. Then, we discuss the language modeling data used in the system. Finally, we give experimental results for various systems.

3.1. Evaluation Metric

We measure the recognition accuracy of numeric entities such as numbers, times, and dates by evaluating a metric similar to the word error rate (WER), but specific to numeric entities. We call this metric the number error rate (NER). To compute the NER, we first align the recognition hypothesis with the reference transcription. We then assume that any word containing a digit is a numeric entity, and we define the NER as follows:

$$\text{NER} = \frac{\text{ND} + \text{NI} + \text{NS}}{\text{NN}}$$

where ND, NI, NS are the numbers of numeric deletions, insertions, and substitutions, respectively. NN is the total number of numeric entities in the reference.

3.2. Baseline System

The baseline speech recognition engine is a standard large-vocabulary recognizer, with PLP features and LDA. The acoustic models use decision trees to train GMM-based triphone HMMs. The acoustic models are trained with maximum likelihood, followed by boosted MMI [11].

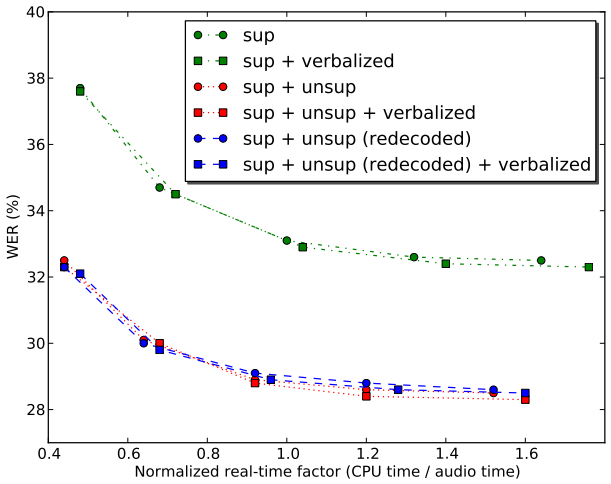

Fig. 4. Word error rate at various normalized real-time factors.

The language model is a standard 5-gram language model trained from web documents and anonymized Google web search queries. Anonymized speech logs are added to the training corpora in some of the experiments below. In all cases, the language modeling pipeline trains a model for each data source, and then linearly interpolates the individual models using a development set to optimize the interpolation weights. The final model is entropy-pruned to a size that fits our production resource targets.

3.3. Experimental Results

We used a set of rewrite grammar rules to expand numeric entities in French. These rules were used to build a verbalizer transducer that can generate verbal expansions for digit sequences, dates, times, postal codes, decimal numbers, cardinal numbers, and ordinal numbers.

The resulting recognizer was evaluated on an anonymized and randomly selected test set that matches our current speech traffic patterns in France. The test set contains 22K utterances and 85K words, of which 2K are numeric entities. The set was transcribed in written language, i.e. the transcribers were instructed to type "23" if they heard "vingt-trois".

We evaluate three different systems. In the first system, (*supervised*), we only used typed text data from web documents and search queries to train the language model. In the second system, (*supervised + unsupervised*), we added anonymized speech recognition logs to the language model training data. These logs consist of recognition results from the then-deployed system, untranscribed, but filtered by recognition confidence. In the third system, (*supervised + unsupervised (redecoded)*), we redecoded the speech logs using the verbalized system from the second system to generate better transcripts. The aim of redecoding is to fix formatting errors on numeric entities caused by the original production system.

We report the performance improvement brought by the verbalizer on these three systems.

Figure 4 shows the word error rate of the evaluated systems for various real-time factors obtained by changing the beam width of the decoder. This experiment clearly shows that using the unsupervised data for language modeling improves the speech recognition

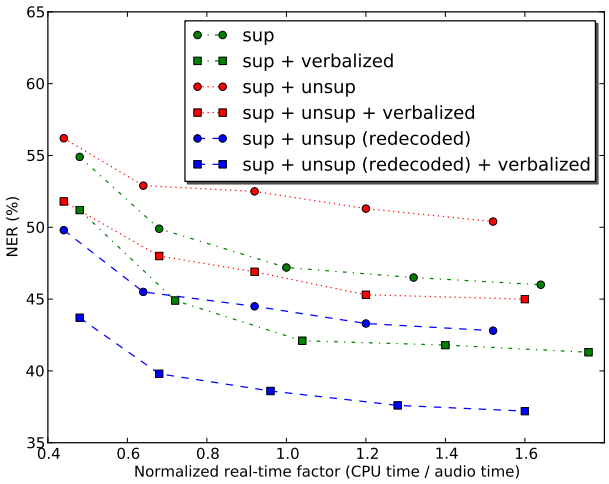

Fig. 5. Number error rate at various normalized real-time factors.

accuracy significantly. For instance, the accuracy improves by about 12% relative at the normalized real-time recognition speed. We can also see that language model verbalization and redecoding the unsupervised data with the verbalized system does not have a significant effect on the WER. This is expected since these experiments concentrate on improving the numeric entity recognition, and the numeric entities constitute a small percentage of the test set. For example, if we removed all the numeric entities from the speech recognition vocabulary, the WER would only degrade about 2%.

The impact of verbalization is obvious by tracking the number error rate, as in Figure 5. This figure clearly shows that language model verbalization significantly improves the recognition accuracy of numeric entities for all the systems. For instance, the best system (*supervised + unsupervised (redecoded) + verbalized*) improves the NER by about 27% over the baseline system (*supervised + unsupervised*) at the normalized real-time recognition speed. It is also interesting to observe that while using unsupervised data improves the WER, it degrades the NER significantly. This is clearly due to a high number of recognition/formatting errors in numeric entities in the baseline system. This shows that using unsupervised data for language modeling may pollute the training data with incorrectly transcribed entities. This data can poison the system to generate more recognition errors for these types of entities even though the WER of the system improves. Redecoding was indispensable in this case.

4. CONCLUSION

We described a transducer-based method to handle non-lexical entities in a speech recognition system. Using a finite-state model to integrate the verbalization into the decoder network provides some advantages. By facilitating the training of the language model in the written domain, it prevents the error-prone conversion of written text to the verbal domain with text-normalization rules. The written-domain language model provides disambiguation power that simple text-normalization rules can not include.

We showed that the language model can be efficiently verbalized by simply composing it with the verbalizer transducer. The resulting verbalized language model converts verbal word sequences to written-domain word sequences. This composition can be seen as verbalizing the language model by converting the written-domain language model into a transducer whose input labels are verbal expansions of the output labels in the grammar. The approach allows multiple ways of verbalizing a vocabulary item, and provides the flexibility to insert intra-word silences where desired. The verbalizer simplifies the lexicon, since the verbalization is separated from the pronunciation modeling, and the lexicon only has to deal with the verbal word forms.

This approach improved the recognition of numerical entities in French by almost 30% over the baseline production system, while maintaining its latency characteristics.

5. ACKNOWLEDGEMENT

We would like to thank Richard Sproat and Martin Jansche for their contributions to rewrite grammar library and rules, and Eugene Weinstein for his help on integration to the infrastructure.

6. REFERENCES

[1] C. Chelba, J. Schalkwyk, T. Brants, V. Ha, B. Harb, W. Neveitt, C. Parada, and P. Xu, "Query language modeling for voice search," in *Spoken Language Technology Workshop (SLT), 2010 IEEE*, dec. 2010, pp. 127 –132.

[2] Maria Shugrina, "Formatting time-aligned ASR transcripts for readability," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Stroudsburg, PA, USA, 2010, HLT '10, pp. 198–206, Association for Computational Linguistics.

[3] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, Dec. 1992.

[4] Cyril Allauzen, Mehryar Mohri, and Brian Roark, "Generalized algorithms for constructing statistical language models," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 2003, ACL '03, pp. 40–47, Association for Computational Linguistics.

[5] Mehryar Mohri, Michael Riley, Don Hindle, Andrej Ljolje, and Fernando Pereira, "Full expansion of context-dependent networks in large vocabulary speech recognition," in *Proceedings of ICASSP 98*, 1998, pp. 665–668.

[6] Mehryar Mohri and Michael Riley, "Integrated context-dependent networks in very large vocabulary speech recognition," in *Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech '99*, 1999, pp. 811–814.

[7] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri, "Openfst: a general and efficient weighted finite-state transducer library," in *Proceedings of the 12th international conference on Implementation and application of automata*, Berlin, Heidelberg, 2007, CIAA'07, pp. 11–23, Springer-Verlag.

[8] Mehryar Mohri and Richard Sproat, "An efficient compiler for weighted rewrite rules," in *34th Annual Meeting of The Association for Computational Linguistics*, 1996, pp. 231–238.

[9] Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai, "The opengrm open-source finite-state grammar software libraries," in *Proceedings of the ACL 2012 System Demonstrations*, Jeju Island, Korea, July 2012, pp. 61–66, Association for Computational Linguistics.

[10] Cyril Allauzen, Michael Riley, and Johan Schalkwyk, "A generalized composition algorithm for weighted finite-state transducers," in *Proceedings of Interspeech*, 2009, pp. 1203–1206.

[11] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for model and feature-space discriminative training," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 2008, pp. 4057 –4060.