Written-Domain Language Modeling for Automatic Speech Recognition

Haşim Sak, Yun-hsuan Sung, Françoise Beaufays, Cyril Allauzen

Google

{hasim,yhsung,fsb,allauzen}@google.com

Abstract

Language modeling for automatic speech recognition (ASR) systems has been traditionally in the verbal domain. In this paper, we present finite-state modeling techniques that we developed for language modeling in the written domain. The first technique we describe is for the verbalization of written-domain vocabulary items, which include lexical and non-lexical entities. The second technique is the decomposition–recomposition approach to address the out-of-vocabulary (OOV) and the data sparsity problems with non-lexical entities such as URLs, e-mail addresses, phone numbers, and dollar amounts. We evaluate the proposed written-domain language modeling approaches on a very large vocabulary speech recognition system for English. We show that the written-domain language modeling improves the speech recognition and the ASR transcript rendering accuracy in the written domain over a baseline system using a verbal-domain language model. In addition, the written-domain system is much simpler since it does not require complex and error-prone text normalization and denormalization rules, which are generally required for verbal-domain language modeling.

Index Terms: language modeling, written-domain, verbalization, decomposition, speech recognition

1. Introduction

Automatic speech recognition systems transcribe utterances into written language. Written languages have lexical entities (e.g. "book", "one") and non-lexical entities (e.g. "12:30", "google.com", "917-555-5555"). The form of the linguistic units output from an ASR system depends on the language modeling units. Traditionally, the language modeling units have been the lexical units in verbal form. The reason for that is we need the pronunciations of the language modeling units for the phonetic acoustic models. Therefore, the common approach has been to pre-process the training text with text normalization rules. The pre-processing step expands the non-lexical entities such as numbers, dates, times, dollar amounts, URLs (e.g. "$10") into verbal forms (e.g. "ten dollars"). With this verbal-domain language modeling approach, the speech recognition transcript in verbal language needs to be converted into a properly formatted written language to present to the user [1, 2]. However, this approach presents some challenges. The pre-processing of training text and the post-processing of the speech transcript are ambiguous tasks in the sense that there can be many possible conversions [3].

An alternative approach, though not common, is written-domain language modeling. In this approach, the lexical and non-lexical entities are the language modeling units. The pronunciation lexicon generally handles the verbalization of the non-lexical entities and providing the pronunciations. One advantage of this approach is that the speech transcripts are in written language. Another advantage is that we benefit from the disambiguation power of the written-domain language model to choose the proper format for the transcript. However, this approach suffers from OOV words and data sparsity problems since the vocabulary has to contain the non-lexical entities.

In this paper, we propose a written-domain language modeling approach that uses finite-state modeling techniques to address the verbalization, OOV and data sparsity problems in the context of non-lexical entities.

2. Written-Domain Language Modeling

We need solutions for two problems to build a language model on written text without first converting to the verbal domain. The first problem is the verbalization of the written-domain vocabulary items, which can be lexical or non-lexical entities. The pronunciations for the lexical entities can be easily looked up in a dictionary. On the other hand, the non-lexical entities are more complex and structured open-vocabulary items such as numbers, web and e-mail addresses, phone numbers, and dollar amounts. For the verbalization of the non-lexical entities, we build a finite-state transducer (FST) as briefly described in section 2.1. [1] The second problem is the OOV words and data sparsity problems for the non-lexical entities. For this problem, we propose the decomposition–recomposition approach as described in section 2.2

2.1. Verbalization

We previously proposed a method to incorporate verbal expansions of vocabulary items into the decoding network as a separate model in addition to the context-dependency network C, the lexicon L, and the language model G, which are commonly used in weighted FST (WFST) based ASR systems [3]. For this purpose, we construct a finite-state verbalizer transducer V so that the inverted transducer V^{-1} maps vocabulary items to their verbal expansions. 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 to their verbal forms. These grammars rely on regular expressions and context-dependent rewrite rules, and are commonly used for text pre-processing and verbal expansion for text-to-speech and text pre/post-processing for speech recognition. They can be efficiently compiled into FSTs [4, 5]. The verbalization model V^{-1} effectively transforms written non-lexical items into lexical items that can be looked up in the lexicon. The approach maintains the desired richness of a written-domain language model, together with the simplicity of a verbal-domain lexicon.

[1] The verbalization approach as applied in a French ASR system will be presented in the ICASSP conference [3]. We describe it here briefly for the sake of completeness and clarity of the explanation for the extended application of it in an English ASR system.

1: $\mathcal{T} \leftarrow$ training corpus
2: $\mathcal{L} \leftarrow$ vocabulary of static pronunciation lexicon
3: $C \leftarrow$ context-dependency model
4: $\mathcal{V} \leftarrow$ vocabulary of $\mathcal{T}$
5: $\mathcal{D} \leftarrow$ FST for decomposition rewrite rule
6: $\mathcal{R} \leftarrow$ a set of FSTs for verbalization rewrite rules
7: $\mathcal{S} \leftarrow$ *build_segmenter_model* ($\mathcal{T}$, $\mathcal{L}$)
8: $\mathcal{M} \leftarrow \emptyset$
9: **for all** $v \in \mathcal{V}$ **do**
10: $d \leftarrow$ *rewrite* (v, $\mathcal{D}$)
11: **if** $d \neq \epsilon$ **then**
12: $d \leftarrow$ *segment_composite_words* (d, $\mathcal{S}$)
13: $\mathcal{M}[v] \leftarrow$ *mark_tokens* (d)
14: **else**
15: $\mathcal{M}[v] \leftarrow v$
16: **end if**
17: **end for**
18: $\mathcal{T}' \leftarrow$ *decompose_corpus* ($\mathcal{T}$, $\mathcal{M}$)
19: $G_d \leftarrow$ *train_language_model* ($\mathcal{T}'$)
20: $R \leftarrow$ *build_restriction_model* ($\mathcal{M}$) (see Figure 2)
21: $V \leftarrow$ *build_verbalization_model* ($\mathcal{V}$, $\mathcal{R}$)
22: $L \leftarrow$ *build_pronunciation_model* ($\mathcal{L}$)
23: $N \leftarrow C \circ L \circ V \circ Proj(R \circ G_d)$

Figure 1: Pseudocode to build the decoding graph for the written-domain language modeling.

$\mathcal{M} \leftarrow$ associative array mapping vocabulary tokens to segmented tokens {e.g. "nytimes.com" $\rightarrow$ "[url] ny$\sim$ times$\sim$ dot$\sim$ com$\sim$ [/url]"}
$w \leftarrow$ compensation cost for the language model probability estimation $P([\text{marker}]|\text{context})$
$n \leftarrow 0, Q \leftarrow I \leftarrow F \leftarrow \{0\}$
for all $(t, s) \in \mathcal{M}$ **do**
  if $t = s$ **then**
    $E \leftarrow E \cup \{0, t, t, 0, 0\}$
  else
    $S \leftarrow tokenize(s)$ {S is list of token segments}
    $b \leftarrow pop\_front(S), e \leftarrow pop\_back(S)$
    $q \leftarrow state[b]$
    if $q = -1$ **then**
      $q \leftarrow state[b] \leftarrow n \leftarrow n + 1$
      $Q \leftarrow Q \cup \{q\}$
      $E \leftarrow E \cup \{0, b, b, -w, q\} \cup \{q, e, e, w, 0\}$
    end if
    for all $s \in S$ **do**
      $E \leftarrow E \cup \{q, clear\_marker(s), s, 0, q\}$
    end for
  end if
end for
$R \leftarrow Determinize(Q, I, F, E)$

Figure 2: Pseudocode for the construction of the restriction–recomposition model R.

2.2. Decomposition–Recomposition

The verbalization model does not solve the OOV words and data sparsity problems for the non-lexical entities. For instance, even with a language model vocabulary size of 1.8 million, the OOV rate for the web addresses is 32% as calculated over the web addresses in a voice search test set. Modeling such in-vocabulary entities as a single unit suffers from the data sparsity problem. Moreover, the verbalization model does not address the pronunciation of composite tokens, e.g. "nytimes.com". We present an approach to model these entities better and to alleviate these problems. Our approach is based on the decomposition of these entities into the constituting lexical units, while offering a method to combine these units back in the FST framework.

The pseudocode for building the decoding graph in the written-domain language modeling is given in Figure 1. The decomposition transducer $\mathcal{D}$ is compiled from a set of rewrite grammar rules. These rules are implemented to decompose non-lexical entities and add special tokens to mark the begin and end of the decomposed segments. For instance, the rewrite grammar rule that we use for URLs decomposes "nytimes.com" to "[url] *nytimes dot com [/url]". This rewrite grammar rule also marks the tokens that might be a composite token with a special symbol (*). These marked composite tokens require further processing to find correct pronunciation.

We build a statistical model for segmenting composite tokens on line 7. The segmentation of the composite tokens is needed to give a proper pronunciation using the pronunciation lexicon L. For this purpose, we train a unigram language model G_s over the vocabulary of the static pronunciation lexicon. Then, we construct an FST S, so that the inverted transducer S^{-1} maps the vocabulary symbols to their character sequences. The composition of two models, $S \circ G_s$ is a weighted FST that can be used for segmenting the composite words. To accomplish that, we simply construct an FST model T for the character sequences of an input word, compose with the segmentation model ($T \circ S \circ G_s$), find the shortest path and print the output labels.

The decomposition transducer $\mathcal{D}$ is used to decompose each token in the vocabulary $\mathcal{V}$ on line 10. If the token is decomposed, we try to segment the decomposed tokens marked with the special symbol (*) using the statistical segmentation model $\mathcal{S}$ on line 12. For the URL example, the segmented tokens will be "[url] ny times dot com [/url]", since for the "nytimes" the most likely segmentation will be "ny times".

We mark each token segment except the marker tokens with a special symbol ($\sim$) on line 13 to differentiate them from the other tokens in the training corpus. We store the segmentation for each token in the vocabulary. For the example, the segmented and marked tokens will be "[url] ny$\sim$ times$\sim$ dot$\sim$ com$\sim$ [/url]". If the token cannot be decomposed with the decomposition transducer, we store the token itself as the segmentation. Using the stored segmentations $\mathcal{M}$ of the tokens in the vocabulary, we decompose the training corpus $\mathcal{T}$ to obtain $\mathcal{T}'$ on line 18. Then, we train an n-gram language model over the decomposed corpus $\mathcal{T}'$ and it is efficiently represented as a deterministic weighted finite-state automaton [6] G_d on line 19.

We construct a finite-state restriction–recomposition model R using the token segmentations on line 20. The pseudocode for the construction of R is given in Figure 2. This algorithm constructs a WFST $R = (Q, I, F, E)$, where Q is a finite set of states, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, E is a finite set of transitions $\{p, i, o, w, q\}$, where p is the source state, i is the input label, o is the output label, w is the cost of the transition, and q is the target state.

An example restriction model for a toy vocabulary of "world", "news", "times", "nytimes.com" with the URL decomposition is shown in Figure 3. The start state 0 maps all the regular words to themselves. We add the special begin marker

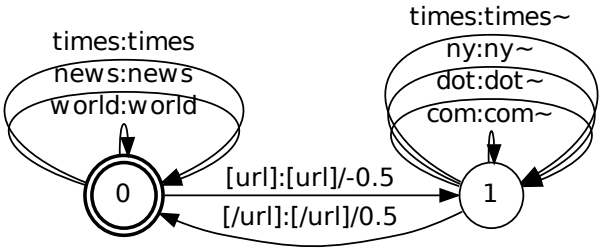

Figure 3: Example restriction–recomposition model R for a toy vocabulary of "world", "news", "times", "nytimes.com".

"[url]" as a transition label to a new state (1) and add the special end marker "[/url]" as a transition label to the start state (0). We add the transition label with the input decomposed segment and the output decomposed segment marked with a special symbol ($\sim$) at this state for each decomposed segment. We can optionally add some rewards and costs to the special marker transitions as shown in Figure 3 to compensate the language model probability estimation for the special begin marker $P([\text{url}]|\text{context})$.

On line 21, we build a verbalization model V using the vocabulary $\mathcal{V}$ and a set of FSTs $\mathcal{R}$ for verbalization rewrite rules. We build a finite-state pronunciation lexicon L on line 22.

The final step on line 23 constructs the decoding graph N. The restriction model R and the language model G_d are composed to get the restricted language model, $R \circ G_d$. This restriction guarantees that the paths in the language model starting with the special begin marker token end with the special end marker token. This is required to get the boundaries of the segmented tokens, so that we can use a simple text processing step to combine the segments and construct the proper written form for these entities. The restricted language model is projected on the input side to obtain the final restricted language model $\text{Proj}(R \circ G_d)$ without the marking symbol ($\sim$). Then, we compose with the verbalizer V, the lexicon L, and the context dependency model C to get the decoding graph N. Note that, the segmented tokens can contain non-lexical entities such as numbers. Therefore, the decomposition approach still depends on the verbalization model for the verbalization of these entities. For instance, the segmented URLs can contain numbers and the verbalization model can provide all the alternative verbalizations for them.

With the proposed approach, the speech recognition transcripts contain the segmented forms for the non-lexical entities that we choose to decompose. However, the begin and end of these segments are marked with the special tokens. Therefore, we apply a simple text denormalization to the transcripts to combine these segments and remove the special tokens. For instance, a possible transcript with this approach will be "go to [url] ny times dot com [/url]" and it will be simply normalized to "go to nytimes.com".

The segmentation of the non-lexical entities alleviates the data sparsity problem. In addition, it addresses the OOV problem for these entities, since the language model trained over the segments can generate unseen entities by combining segments from different entities.

3. Systems & Evaluation

Our acoustic models are standard 3-state context dependent (triphone) HMM models which use a deep neural network (DNN) to estimate HMM-state posteriors [7]. The DNN model is a standard feed-forward neural network with 4 hidden layers of 2560 nodes. The input layer is the concatenation of 26 consecutive frames of 40-dimensional log filterbank energies calculated on 25ms windows of speech every 10ms. The 7969 softmax outputs estimate the posterior of each state. We use 5-gram language models pruned to 23 million n-grams using Stolcke pruning [8]. An FST-based search [9] is used for decoding.

We measure the recognition accuracy of numeric entities such as numbers, times, dollar amounts, and web addresses by evaluating a metric similar to the word error rate (WER). We specifically split the entities to two different groups, numeric and web address. We call this metric the entity error rate (EER). To compute the EER, we first remove all the tokens not matching the entity type from the recognition hypothesis and the reference transcript, then calculate the standard word error rate over the remaining entities.

3.1. Baseline Verbal-Domain System

The language model used for the baseline verbal-domain system is a 5-gram verbal-domain language model obtained by Bayesian interpolation technique [10]. A dozen individually-trained Katz-backoff n-gram language models from distinct data sources in verbal domain are interpolated. The language models are pruned using Stolcke pruning [8]. The sources include typed data sources (such as anonymized web search queries and SMS text) and unsupervised data sources consisting of ASR results from anonymized utterances which have been filtered by their recognition confidence score. The data sources used vary in size, from a few million to a few billion sentences, making a total of 7 billion sentences. The vocabulary size of this system is 2 million.

In the verbal-domain system, the web addresses are handled by using some text normalization and denormalization FSTs for splitting the web addresses to lexical entities in the language model training data and combining the lexical entities in the speech recognition transcript to form a web address if it is in the list of known web addresses.

3.2. Written-Domain System

The written-domain language model was trained using similar data sources and techniques to the baseline system but in written domain. The unsupervised data source of speech recognition transcripts in written domain was obtained by redecoding anonymized utterances. For redecoding, we used an initial written-domain language model trained on SMS text, web documents, and search queries. We applied simple text normalizations to clean up the training text (e.g. "8 pm" $\rightarrow$ "8 p.m."). We also filtered the speech recognition transcripts by using their recognition confidence scores. The unsupervised transcripts provide domain adaptation for the final language model. We used all the data sources to train the final language model. The vocabulary size of this system is 2.2 million.

For the written-domain system, we used a set of rewrite grammar rules to expand entities including numbers, time, and dollar amounts in English into verbal forms. These rules were used to build the verbalization transducer that can generate verbal expansions for digit sequences, times, postal codes, decimal numbers, cardinal numbers, ordinal numbers, and time. Table 1

Table 1: *A list of rewrite grammar rules with examples for verbalization.*

Rule	Written Form	Verbal Form
Cardinal	2013	two thousand thirteen
Digit	2013	two zero one three
Two-digit	2013	twenty thirteen
Ordinal	23rd	twenty third
Time1	3:30	three thirty
Time2	3:30	half past three
Dollar1	$3.30	three dollars thirty cents
Dollar2	$3.30	three thirty dollars

Table 2: *Word error rates for verbal-domain and written-domain systems on three test sets.*

	Verbal-Domain(%)	Written-Domain(%)
Search	32.1	32.1
Mail	8.3	7.8
Unified	12.5	12.0

shows a simplified list of verbalization grammar rules.

We focus on improving recognition accuracy for web addresses and phone numbers. We used simple rewrite grammar rules to decompose web addresses ("google.com" → "[url] *google dot com [/url]") and phone numbers ("555-5555" → "[phone] 5 5 5 55 55 [/phone]"), as described in section 2.2.

3.3. Experimental Results

The systems were evaluated on three anonymized and randomly selected test sets that match our current speech traffic patterns in English. The first test set – *Search* has 41K utterances and consists of voice search utterances. The second one – *Mail* has 18K utterances and consists of e-mail dictation utterances. The final one – *Unified* has 23K utterances and is a unified set of voice search and dictation utterances. All test sets are hand-transcribed in written domain and we measure the speech transcription accuracy in written domain.

The baseline verbal-domain system uses a set of denormalization FSTs to covert recognition transcripts in verbal form to corresponding written forms (e.g. "ten thirty p.m."→"10:30 p.m."). Table 2 shows the performance of the baseline verbal-system on the *Search*, *Mail*, and *Unified* test sets. Without the denormalization FSTs, the performance drops significantly because of the verbal and written mismatch. For instance, on the *Unified* test set the word error rate increases from 12.5% to 13.9% without the denormalization.

In the written-domain system, there is no text denormalization rule applied to the recognition transcript except a simple one to combine the token segments marked clearly as discussed in section 2.2. As Table 2 shows, the written-domain system outperforms the verbal-domain system by 0.8% and 0.7% on *Mail* and *Unified* test sets. Because some entities have ambiguities in verbal-to-written conversion and require context to distinguish, using verbalizer and decomposer in the language model provides context to resolve the ambiguities. However, the text denormalization rules used in verbal-domain system are completely independent of the language model and have issues to resolve these ambiguities.

We specifically look at recognition results on numeric entities (numbers, dollar amounts, phone numbers, times) and URL entities (web addresses) and report entity error rates in Table 3.

Table 3: *Entity error rates for numeric and URL entities of verbal-domain and written-domain systems.*

	Verbal-Domain(%)	Written-Domain(%)
Numeric	68.9	59.5
URL	57.7	54.1

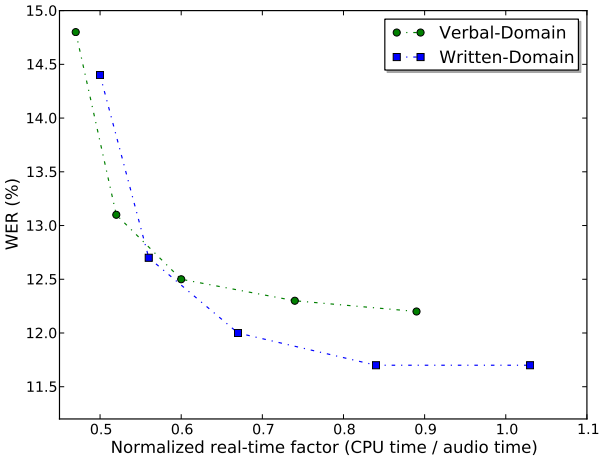

Figure 4: WER at various normalized real-time factors.

We use the *Search* test set for these experiments. There are 2525 numeric entities with only 18 OOV numeric entities (0.7%) and 1202 URL entities in the *Search* test set. There are no OOV URL entities since we use the decomposition–recomposition approach to decompose the URLs. If we don't use this approach, the OOV rate for URL entities is 32% as calculated over the URL entities in the test set. The written-domain system performs better than the verbal-domain system for both numeric error rate and URL error rate.

Figure 4 shows the word error rate of the systems for various real-time factors obtained by changing the beam width of the decoder. Both systems can improve accuracy further by sacrificing speed. The saturated performance of the written-domain system is better than that of the verbal-domain system.

4. Conclusion

We presented two techniques for the written-domain language modeling in the finite-state transducer framework. The verbalization and the decomposition–recomposition techniques work together to address the verbalization, OOV words, and data sparsity problems in the context of non-lexical entities. The written-domain language modeling using the proposed approaches overcomes the shortcomings of the verbal-domain language modeling. First of all, it simplifies the speech recognition system by eliminating complex and error-prone text normalization and denormalization steps. Secondly, it significantly improves the speech transcription accuracy in written language, since we receive an advantage from the contextual disambiguation of the written-domain language model. Finally, the decomposition–recomposition approach together with the verbalization model provides an elegant and contextual language model integrated solution for the pronunciation and modeling of non-lexical entities.

5. 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] M. 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*, ser. HLT '10. Association for Computational Linguistics, 2010, pp. 198–206.

[3] H. Sak, F. Beaufays, K. Nakajima, and C. Allauzen, "Language model verbalization for automatic speech recognition," in *Acoustics, Speech and Signal Processing, 2013. ICASSP 2013. IEEE International Conference on*, 2013. [Online]. Available: http://goo.gl/xmCOR

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

[5] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The opengrm open-source finite-state grammar software libraries," in *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, July 2012, pp. 61–66.

[6] C. Allauzen, M. Mohri, and B. Roark, "Generalized algorithms for constructing statistical language models," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03. Association for Computational Linguistics, 2003, pp. 40–47.

[7] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proceedings of Interspeech*, 2012.

[8] A. Stolcke, "Entropy-based pruning of backoff language models," in *DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[9] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: a general and efficient weighted finite-state transducer library," in *Proceedings of the 12th international conference on Implementation and application of automata*, ser. CIAA'07. Springer-Verlag, 2007, pp. 11–23.

[10] C. Allauzen and M. Riley, "Bayesian language model interpolation for mobile speech input," in *Proceedings of Interspeech*, 2011, pp. 1429–1432.