

ON EFFICIENT COUPLING OF ASR AND SMT FOR SPEECH TRANSLATION

Bowen Zhou, Laurent Besacier and Yuqing Gao

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

ABSTRACT

This paper presents an efficient tightly integrated approach for improved speech translation performance. The proposed approach combines the automatic speech recognition (ASR) and statistical machine translation (SMT) components in a bi-directional fashion. First, our SMT decoder takes the speech recognition lattice to perform an integrated search for the optimal translation by combining various ASR scores and translation models. Our approach is implemented within the recently proposed Folsom SMT framework that employs a multilayer search algorithm to conduct efficient operations on multiple graphs, which not only achieves memory efficiency and fast speed that is critical for real time speech translation applications, but also provides significant accuracy improvements. Secondly, we also report our experiments where the ASR is customized by reinforcing the language model to favor downstream translation component. We evaluated our approach on a large vocabulary speech translation task, and we obtain more than 2 point BLEU improvement over standard cascaded 1-best speech translation.

1. INTRODUCTION

Automatic translation of spoken audio streams from one language into another one has been one of most attractive fields in recent years. To accomplish such a task, the system is typically comprised of individual engines for automatic speech recognition (ASR) and machine translation (MT), as well as other downstream components (e.g., speech synthesis or information extraction etc) depending on the requirements of target application.

There are several ways to combine ASR and MT components in a spoken language translation system, and the architectures for such systems can either be cascaded or integrated. In cascaded approaches, the SMT has been conducted on the single best ASR output, i.e., SMT is carried out with the assumption of that the ASR output is perfect. Given the fact that the single best ASR output can not always be correct, and more appropriate recognition results are often embedded in competing hypotheses produced by the recognizer, it is natural to enhance the interface between ASR and MT by extending the single best hypothesis to multiple recognition options. There have been a number of previous studies [1, 2, 3, 4] in this thread that employs either N-best list or recognition lattice to incorporate more information into a translation system. In contrast to the cascaded approach, we refer to the latter as integrated approach.

Ideally for the speech translation tasks, the choice of the best ASR should not be only determined locally by the recognizer, but also by the MT engine to jointly select a globally optimal one that maximizes the translation quality in the target language. There-

fore, it is expected that the speech translation accuracy should be improved by such an integration efforts.

However, there are a number of issues to be addressed for this integration approach. First, the search space of the integrated translation decoding has been dramatically increased by several orders of magnitude compared to the single best or text-based translation, and thus, more efficient decoding algorithms are required to handle the space issue, which is particularly important for any real-time interactive applications such as speech-to-speech translation. Secondly, the relative improvements presented in some of previous studies have been shown to be limited or inconsistent, which does not justify the significant additional computational expenses (CPU time and memory usage) associated with the integration approach. Thirdly, previous studies have been mostly focused on how to incorporate ASR information into MT, and there are few studies related to how MT might be able to help ASR. Finally, while some researchers, such as Matusov et al. [3], have reported consistent improvements on small to medium translation tasks, to our best knowledge, there have been no reported consistent improvements on large vocabulary speech translation tasks. It is our intention to show in this paper that the integration approach should also work for large vocabulary tasks with reasonable speed for real time applications.

In this study, we describe our approach that addresses above issues. Our approach is based on our previous proposed Folsom SMT translation framework [5] that employs a multilayer search algorithm. Within this framework, the integrated lattice translation can be achieved by extending the layer of input, which achieves not only memory efficiency, but also provide significant accuracy improvements compared to the 1-best translation. Furthermore, our proposed approach attempts to combine ASR and SMT components in a bi-directional fashion. To that end, we report our experiments that the ASR is customized by reinforcing the language model to favor downstream translation component. We evaluated our approach on a large vocabulary speech translation task from Iraqi Arabic to English, and we obtain more than 2 point BLEU improvement over standard cascaded 1-best speech translation. Moreover, our implementation within Folsom framework achieves a fast lattice translation speed that is critical to any real-time applications.

The remainder of this paper is organized as follows: Sec. 2 presents an overview of our baseline SMT system for speech translation; Sec. 3 describes details of our new coupling approach and implementation; Sec. 4 presents experimental results for speech translation tasks; and finally, Sec. 5 summarizes our contributions.

2. FOLSOM: AN EFFICIENT PHRASE-BASED SMT FRAMEWORK

Our recently proposed Folsom system [5] is a novel framework for phrase-based SMT that is specifically designed to achieve efficiency and fast translation speed, yet to maintain state-of-the-art translation

We thank the DARPA TRANSTAC program for providing funding for this work.

accuracy. We provide a brief overview of the Folsom framework in this section, and details have been presented in [5]. The Folsom framework has additional advantages for speech translation tasks, as it provides a convenient combination of speech recognition and machine translation, which will be shown in Sec. 3.

Phrase-based translation models explicitly take word contexts into consideration when making a translation decision. The foreign word sequence f_1^J is segmented into phrase sequence \bar{f}_1^K , where $1 \leq K \leq J$. We express the translation model as a chain of conditional probabilities, which can be represented by finite-state machines (FSM's) that model the relationships between inputs and outputs [5]. Therefore, the translation task can be framed as finding the best path in the following FSM:

$$S = \mathcal{I} \circ \mathcal{P} \circ \mathcal{T} \circ \mathcal{W} \circ \mathcal{L} \quad (1)$$

where \mathcal{I} denotes the source word sequence expressed as a linear finite-state automaton, the transducers \mathcal{P} , \mathcal{T} , \mathcal{W} , and \mathcal{L} correspond to source language segmentation transducer, phrase translation transducer, target language phrase-to-word transducer, and target language model, respectively.

For extracting bilingual phrase pairs and estimating their translation probabilities, we follow the procedure described in [6]. The bilingual phrase pair inventory, denoted as \mathcal{BP} , is extracted from the union of bidirectional word-level alignments. The phrase translation probabilities are estimated via maximum likelihood estimation using the counts derived from \mathcal{BP} .

2.1. Transducers for Component Models

Source Language Segmentation transducer \mathcal{P} : It explores all ‘‘acceptable’’ phrase sequences for any given source sentence. We assume a uniform distribution over all acceptable segmentations. Within the FSM framework, the segmentation procedure is implemented as a transducer \mathcal{P} that maps from word sequences to phrases. In general, this transducer is not determinizable due to the overlap between phrases [7, 5]. In our work, we introduce an auxiliary symbol, denoted \mathcal{EOP} , marking the end of each distinct source phrase. Once we have determinized the transducer, we can replace the \mathcal{EOP} markers with empty strings in a later step as appropriate. As we assume a uniform distribution over segmentations, we simply set the cost (or negative log probability) associated with each arc to be zero.

Phrase translation transducer \mathcal{T} : The phrase translation model is implemented by a weighted transducer that maps source phrases to target phrases. In order to be consistent with the other transducer's in Eq. (1), one more arc is added in this transducer to map \mathcal{EOP} to itself with no cost.

Target language phrase-to-word transducer \mathcal{W} : To incorporate target language model during translation, the target phrases must be converted back to target words. It is clear that the mapping from phrases to word sequences is deterministic. Therefore, the implementation of this transducer is straightforward. Again, we need to place the auxiliary token \mathcal{EOP} on additional arcs to mark the ends of phrases.

Target language model \mathcal{L} : The target language model can be represented by a weighted acceptor that assigns probabilities to target language word sequences based on a back-off N-gram language model [7].

2.2. The Multilayer Search Algorithm

The translation problem can be framed as finding the best path in the lattice S for a given input automaton \mathcal{I} , as shown in Eq. 1. Viterbi search can therefore be applied to S to find its lowest-cost

path. To minimize the amount of computation required at translation time, it is desirable to perform as many composition operations in Eq. (1) as possible ahead of time. The ideal situation is to compute $\mathcal{H} = \mathcal{P} \circ \mathcal{T} \circ \mathcal{W} \circ \mathcal{L}$ offline.

However, it can be very difficult to construct \mathcal{H} given practical memory constraints for large translation tasks. In particular, the nondeterministic nature of the phrase translation transducer interacts poorly with the language model. Furthermore, even when one is able to compute and store \mathcal{H} , the composition $\mathcal{I} \circ \mathcal{H}$ itself may be quite expensive. To improve speed, it has been proposed that *lazy* or on-the-fly composition be applied, followed by Viterbi search with beam pruning. Nevertheless, for large \mathcal{H} , using such operations from general FSM toolkits can still be quite slow and memory inefficient.

While it may not be feasible to compute \mathcal{H} in its entirety as a single FSM, it is possible to separate \mathcal{H} into two pieces: the language model \mathcal{L} and the translation model \mathcal{M} , which we call a SIPL:

$$\mathcal{M} = \text{Min}(\text{Min}(\text{Det}(\mathcal{P}) \circ \mathcal{T}) \circ \mathcal{W}) \quad (2)$$

where Min denotes the minimization operation. Due to the determinizability of \mathcal{P} , \mathcal{M} can be computed offline using a moderate amount of memory. We perform all operations using the tropical semiring as is consistent with Viterbi decoding.

To address the problem of efficiently computing $\mathcal{I} \circ \mathcal{M} \circ \mathcal{L}$, we have developed a multilayer search algorithm for the Folsom system. The basic idea is that we perform search in multiple FSM's or layers simultaneously. Specifically, as shown in Fig. 1, we have one layer for each of the input FSM's: \mathcal{I} , \mathcal{L} , and \mathcal{M} . At each layer, the search process is performed via a state traversal procedure starting from the start state s_0 .

In previous work, we represent each state \vec{s} in the search space using the following 7-tuple: $(s_{\mathcal{I}}, s_{\mathcal{M}}, s_{\mathcal{L}}, c_{\mathcal{M}}, c_{\mathcal{L}}, \vec{h}, \vec{s}_p)$, where $s_{\mathcal{I}}$, $s_{\mathcal{M}}$, and $s_{\mathcal{L}}$ record the current state in each input FSM; $c_{\mathcal{M}}$ and $c_{\mathcal{L}}$ record the accumulated cost in \mathcal{L} and \mathcal{M} in the best path up to this point; \vec{h} records the target word sequence labeling the best path up to this point; and \vec{s}_p records the best previous state.

To support platforms without floating-point units (e.g., most handheld devices), search in the Folsom system is implemented using integerized arithmetic. Moreover, memory usage can significantly be reduced by storing the SIPL (\mathcal{M}) and the language model \mathcal{L} on disk and paging them in on demand. We store them in sorted structures that make I/O access very efficient. The runtime memory required by Folsom decoder is mostly devoted to storing active search hypotheses, and it can be as little as 10MB.

3. INTEGRATED SPEECH TRANSLATION IN FOLSOM

3.1. Incorporating ASR Uncertainty into Translation

The design architecture of the Folsom system provides a convenient and efficient way to perform integrated speech translation. To handle the recognition lattice, the source layer in the decoding graph needs to be extended to allow for competing recognition hypotheses at any arbitrary instance. Furthermore, the recognition probabilities of individual hypothesis need to be taken into account in the translation search to make global decisions.

To handle recognition uncertainty, the search hypothesis \vec{s} is extended into the following 8-tuple: $(s_{\mathcal{I}}, s_{\mathcal{M}}, s_{\mathcal{L}}, c_{\mathcal{I}}, c_{\mathcal{M}}, c_{\mathcal{L}}, \vec{h}, \vec{s}_p)$; here, the additional element $c_{\mathcal{I}}$ denotes the accumulated cost in the recognition lattice, which, in our approach, is a combination of scores of acoustic model, language model and fast match. The timing information presented in the recognition lattice is not used in our current integrated decoding, and thus are discarded in the upper layer

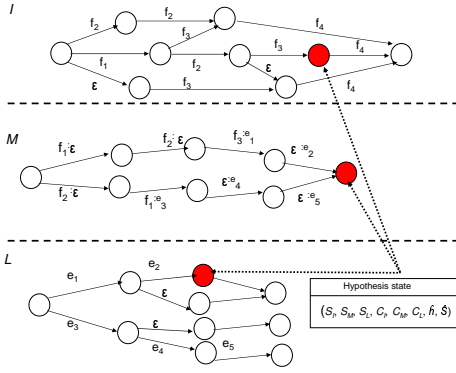


Fig. 1. The multi layer search for lattice translation (for simplicity, the cost associated with each arc is not labeled).

of Fig. 1. Moreover, the recognized out-of-translation-vocabulary entries such as fillers, silence and partial words have been replaced with ϵ transitions since they are irrelevant to translation model.

At initialization, only the start state \vec{s}_0 is active, which corresponds to being located at the start state of each input FSM with no accumulated costs. The active states at each position t are computed from the active states at the preceding position $t - 1$ in the following way: for each active state \vec{s} at position $t - 1$, first advance $s_{\mathcal{I}}$ in recognition lattice. Then, look at all outgoing arcs of $s_{\mathcal{M}}$ labeled with the current input word f , and traverse one of these arcs, advancing $s_{\mathcal{M}}$. Then, given the output label e of this arc, look at all outgoing arcs of $s_{\mathcal{L}}$ with e as its input, and traverse one of these arcs, advancing $s_{\mathcal{L}}$. The set of all states $(s_{\mathcal{I}}, s_{\mathcal{M}}, s_{\mathcal{L}}, \dots)$ reachable in this way is the set of active states at position t . The remaining state components $c_{\mathcal{M}}, c_{\mathcal{L}}, \vec{h}$, and \vec{s}_p must be updated appropriately, and ϵ -transitions in different machines must be handled correctly as well. For each active state, the hypothesis \vec{h} is a translation of a prefix of a given path in the input layer, and can conceivably grow to be quite large. However, we can store the \vec{h} 's for each state efficiently using the same ideas as used in token passing in ASR. In particular, the set of all active \vec{h} 's can be compactly represented using a prefix tree, and each state can simply keep a pointer to the correct node in this tree.

The associated cost of a given active state \vec{s} is computed as:

$$\text{cost}(\vec{s}) = \lambda_{\mathcal{I}} c_{\mathcal{L}} + \sum_i \lambda_i c(i) \quad (3)$$

where, $\lambda_{\mathcal{I}}$ is the weight parameter for scaling the impact of recognition uncertainty, and the second term is the translation cost, which is a log-linear combination of costs computed by multiple models (for example, the translation model, language model as well as other heuristic costs such as translation length penalties and phrase penalties). All the λ 's in Eq. 3 can be determined by maximizing some objective score with respect to a dev set using Powells algorithm combined with a line search method [8].

To reduce the search space, active search states are merged whenever they have identical $s_{\mathcal{I}}, s_{\mathcal{M}}$, and $s_{\mathcal{L}}$ values; the remaining state components are inherited from the state with lower cost. In addition, two pruning methods, histogram and beam pruning, are used to achieve the desired balance between translation accuracy and speed. The set of legal translation candidates are those \vec{h} associated with

Table 1. A comparison of ASR performance (WER %).

Data	Baseline		Customized	
	436K	145K	436K	145K
Dev(1-best)	43.1	48.2	38.7	47.9
Dev(Oracle)	24.2	27.3	22.6	27.3
Test(1-best)	27.5	29.4	25.7	27.7
Test(Oracle)	12.6	15.0	12.7	15.4

states \vec{s} where each component substate is a final state in its layer. The selected candidate is the legal candidate with the lowest accumulated cost.

Our decoder can be viewed as an optimized version of dynamic composition combined with *lazy* determinization and minimization [7], as well as Viterbi search. However, our algorithm has the advantage of not only being possibly much faster and more memory efficient than general composition implementations found in FSM toolkits, but it can also incorporate information sources that cannot be easily or compactly represented using FSM's.

3.2. Customizing ASR for MT

Our baseline ASR system for Iraqi Arabic is more precisely described in [9]. We used the same discriminatively trained acoustic models for all the experiments described in this paper. The acoustic model for dialectal Arabic uses graphemes as the basic acoustic-phonetic units. There are 33 graphemes representing speech and silence. Each phone is modeled with a 3-state left-to-right HMM. The acoustic models are built using 40 dimensional features. The context-dependent model has over 2K leaves and 60K Gaussians.

Towards a more tightly integrated speech translation approach, we propose to further customize the ASR module in the favor of MT. As an initial attempt, we intend to reinforce the source language model (SLM) using knowledge presented in the translation model. The general idea is that, other things being equal, the recognizer should favor recognition hypotheses that the MT engine has the highest confidence to translate, in order to achieve a higher source speech to target text translation performance. One way to approximate this idea is to bias the SLM based on translation model, and more specifically, the phrase-level translation table employed in our phrase-based SMT system.

Practically, for each entry of the phrase translation table, we obtain the foreign part to train a new SLM, and we assign different weights to each foreign phrase, according to the phrasal translation probability. In such a way, the probability of some rare source word sequences that yet have less ambiguous translations will be boosted in this new SLM. It is not surprising to us that this new SLM alone degrades the ASR performance due to its bias, compared to our baseline SLM, as shown in our preliminary experiments. Our final customized LM used in the experiments reported here is actually an interpolation (both weights fixed a priori to 0.5) between the baseline SLM and this biased SLM. All our language model uses standard 3-gram.

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

We evaluate our integrated Folsom speech translation system on a real-time speech translation task translating Iraqi Arabic into English. The SMT training data has a total of 436k parallel sentences, which correspond to about 2.1M running words on the Arabic part

Table 2. Speech translation (BLEU %) on smaller (145k) training data.

Data	Input	Baseline ASR	Customized ASR
Dev	1-best	32.73	34.56
	Integrated	34.95	35.63
	Correct	50.36	
Test	1-best	34.63	35.03
	Integrated	35.65	36.43
	Correct	41.20	

Table 3. Speech translation (BLEU %) on bigger (436k) training data.

Data	Input	Baseline ASR	Customized ASR
Dev	1-best	39.80	39.63
	Integrated	42.62	47.02
	Correct	62.96	
Test	1-best	36.15	36.22
	Integrated	38.30	37.36
	Correct	44.74	

and about 3M on the English part. The vocabulary size is above 75K for Arabic and above 25K for English. To evaluate the effectiveness of our method on various sizes of training data, we also downsampled the data set to obtain a smaller amount of training data with 145k parallel sentences (i.e., 1/3 of the original size). The dev and test data are extracted from different test scenarios in Iraqi Arabic: the dev set is made up of 510 speech utterances coming from 5 speakers; the test set is made up of 554 spoken utterances from 4 speakers, which is in total more than 1 hour in length. For translation evaluation, the dev set provides 4 references while the test set has 1 reference.

The acoustic training data consists of about 200 hours of dialectal Iraqi Arabic collected in the context of the TRANSTAC speech to speech translation project. All the baseline language models used in the experiments reported here use a training corpus which is the Iraqi part of the corresponding parallel data set described above. The maximum phrase length was set to 5 in all our experiments. For the bigger training set, we obtained about 2.2M phrase pairs from the training data. The resulting SIPL has about 4.3M states and 6.5M arcs. For the smaller one, the obtained phrase pairs are reduced to 740K. The English language model in the translation system is fixed in all the experiments, which is a back-off 3-gram trained from 4M monolingual words. All decoding experiments were conducted on a Linux machine with a 2.4 GHz Pentium 4 processor and 512 MB of memory, with \mathcal{M} and \mathcal{L} loaded entirely into memory.

4.2. Experimental Results

We first report our ASR results (1-best WER and lattice oracle WER) in Table 1. The differences observed between 1-best and oracle WER indicate the potential benefits we might have in using lattices inputs to the translation module. We note that the 1-best recognition accuracy improves significantly over the baseline for all scenarios by using our customized language model, while the improvements for oracle WER is less significant. The big improvement of 1-best accuracy is rather unexpected, which might be due to a better smoothing of the customized LM resulting from the interpolation, as described in Sec. 3.2.

Speech translation results are presented in Table 2 and Table 3 in terms of the BLEU metric [10] respectively for two setups. The dev sets are employed to adjust the model weights in Eq. 3 to optimize the BLEU scores for different inputs (i.e., correct transcript, 1-best

ASR output and ASR lattice). These weights are consequently applied to the test set with corresponding inputs.

From both Table 2 and Table 3, we observe consistent improvement of the integrated decoding compared to 1-best on both dev and test set for all scenarios. The BLEU improvement is significant (ranging from 1 to more than 2 points for most cases, and up to 6 points on dev set in Table 3) for all scenarios. This clearly shows that our integrated decoding significantly enhances large vocabulary speech translation task. Table 2 also shows that our customized ASR advances translation performance for all scenarios. Combining with integrated decoding, we achieve 3 points BLEU improvement on dev set and 2 points on test set. However, the benefit of customized ASR is less clear on the bigger training set.

Moreover, we measured translation speeds for our integrated speech translation system. For all lattice decoding experiments, the Folsom system is able to finish the entire test set (approximately 50 mins in length) of 554 lattice in 45 seconds. In other words, our system speed should be more than satisfactory for our real time speech translation tasks.

5. SUMMARY

We presented an efficient and more tightly integrated speech translation system within the Folsom SMT framework. Our approach combines ASR and SMT in a bi-directional fashion by using a recognition lattice and a customized ASR for MT. We evaluated our system on a large vocabulary task and observed consistent improvements over cascaded baseline system. Furthermore, our implementation within the Folsom SMT framework provides fast decoding speed that is critical for real time speech translation.

6. REFERENCES

- [1] R. Zhang et al., “A unified approach in speech-to-speech translation: Integrating features of speech recognition and machine translation,” in *COLING*, 2004.
- [2] S. Saleem et al., “Using word lattice information for a tighter coupling in speech translation systems,” in *ICSLP*, 2004.
- [3] E. Matusov et al., “On the integration of speech recognition and statistical machine translation,” in *Interspeech*, 2005.
- [4] L. Mathias et al., “Statistical phrase-based speech translation,” in *International Conference of Acoustic Speech and Signal Processing*, 2006.
- [5] B. Zhou et al., “Folsom: A fast and memory-efficient phrase-based approach to statistical machine translation,” in *IEEE/ACL Workshop on Spoken Language Technology*, 2006.
- [6] F. J. Och et al., “Improved alignment models for statistical machine translation,” in *Proc. EMNLP/VLC’99*, MD, USA, 1999, pp. 20–28.
- [7] M. Mohri et al., “Weighted finite-state transducers in speech recognition,” *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.
- [8] W. H. Press et al., *Numerical Recipes in C++ : the art of scientific computing*, Cambridge University Press, 2002.
- [9] M. Afify et al., “On the use of morphological analysis for dialectal arabic speech recognition,” in *ICSLP*, 2006.
- [10] K. Papineni et al., “Bleu: a method for automatic evaluation of machine translation,” Technical Report RC22176, IBM T. J. Watson Research Center, 2001.