RAPID INTEGRATION OF PARTS OF SPEECH INFORMATION TO IMPROVE REORDERING MODEL FOR ENGLISH-FARSI SPEECH TO SPEECH TRANSLATION

Sameer Maskey, Bowen Zhou

IBM T.J Watson Research Center
Yorktown Heights, New York
smaskey@us.ibm.com, zhou@us.ibm.com

ABSTRACT

Integrating Parts of Speech (POS) information to Machine Translation (MT) model usually amounts to significant changes in the MT decoder. We present a method to rapidly integrate POS information without adding complexity to the decoder. We show how we can re-estimate the lexicalized reordering probability estimates with POS tags during the training time without having to use POS tagger at the decoding phase. We present our empirical results for two different MT decoding algorithms that use lexicalized reordering models.

Index Terms— Machine Translation, Reordering Model, Speech Translation

1. INTRODUCTION

The current data-driven statistical methods of MT such as IBM Models [1], phrase-based models [2], and syntax-based system [3] all require significant amount of data. The probability estimates for the translation get more robust as the counts of samples for each parameter increases. Problem arises when there is not enough data to estimate the distribution of some of the parameters, particularly when we are trying to translate between languages that have very different word order – English and Persian (Farsi). Let us describe the reordering problem in English and Farsi.

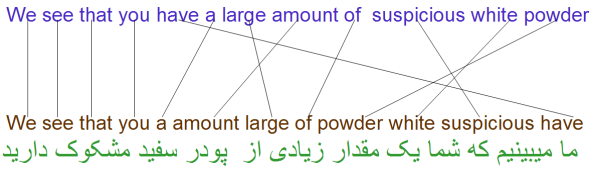

Fig. 1. Reordering Differences in English and Farsi

English is a language where most of the sentences are constructed in Subject (S) Verb (V) Object (O) order. For example in Figure 1 we can see that object (powder) comes at the end of the English sentence. On the other hand Farsi sentences are constructed in Subject (S) Object (O) Verb (V) order. In Farsi sentence shown in Figure 1 we see that verb "have" is at the end of the sentence. Let us describe some of the details of this word-ordering problem in further detail below.

Regular and Modal Verbs: In Farsi, verbs and modal verbs tend to appear at the end of the sentences; but sometimes modal verbs remain at the same position when main verb appears at the end of the sentence or vice versa. This makes the reordering of verbs very complicated. In the example in Table 1 modal verb 'can' does not move while the example in Figure 1 the modal verb 'have' moved to the end of the sentence.

She **can** speak English
She **can** English speak

Table 1. Modal and Regular Verb Reordering

Noun + **Adj** In Figure 1 the adjectives suspicious and white are swapped with the noun (powder) in Farsi. This is a common structure in Farsi where adjective and the noun it modifies are swapped. Another example is shown in Table 2 where the phrase 'blue pen' is swapped to 'pen blue' in Farsi.

I lost my **blue** pen.
I pen **blue** my lost.

Table 2. Reordering of Adjectives

We can note from the above examples that Farsi word order is significantly different from English. We should also note though that there is some consistent pattern in the way adjectives, verbs and nouns are reordered. We would like to exploit the pattern of movements of words based on POS to improve the reordering model.

2. RELATED WORK

There has been a lot of work on trying to improve the reordering model for a machine translation system. The phrase based translation system [2] was a significant development in MT because the model was able to better estimate local reordering better than the IBM models [1].

[4] introduced a lexicalized block reordering model where two consecutive phrases can be swapped. The swapping of phrases allowed words to be reordered longer distances. [5] proposed a distortion based model that allowed words to move any distance within the maximum distortion window. The distortion distance probabilities were computed using the word level alignment.

[6] proposed syntax based MT that differs significantly from models described above because the reordering was dependent on lexical information plus the syntactic information obtained from constituent parse trees. Another set of approaches that have been used is to use only the syntax information on the source side. [7] reorders clauses on the source side by using clause structure.

Even though all of the above methods are viable ways to improve the reordering model, many of the methods may not be feasible for us due to the type of constraints we face as one of the participating teams of the Transtac program.

3. IMPROVING REORDERING MODEL WITHIN CONSTRAINTS

Transtac[1] is Darpa funded program where competing teams are given a certain length of time to develop a Speech-to-Speech (S2S) system for given language pairs. Due to the nature of the task any new MT algorithm or improvement we make have to be viable under the constraints we describe below.

First, we have a constraint of memory used by the MT engine. We need to fit all the components of S2S system, ASR, MT and synthesis for both languages in memory (approximately 1GB). In order to fit so many components in 1GB of memory, we need to make sure MT engine takes as less amount of memory as possible.

Second, we have a constraint of latency, i.e. our MT engine is tested as a part of speech-to-speech translation system for conversation so we cannot add much CPU time to the decoder running on 1GHz portable machine. Hence running CPU heavy parsers during decoding may not be feasible.

Third, we have a time constraint of having to build a speech-to-speech translation system in a short amount of time (a few months). Hence rapid development of the MT engine for new language pairs is essential.

We need to improve the reordering model with these constraints in our setup. We propose a method that does not add complexity to the decoder. It also does not require any new knowledge resource during the decoding time so does not consume any new memory. Since the changes are only in the training phase no change in APIs for decoder is needed, so rapid development is possible. We first describe our baseline models and decoders in the next section.

4. BASELINE REORDERING MODELS AND DECODERS

4.1. Decoder One (D1): Stack Based Decoder

We performed experiments on two of our MT engines that use different types of decoders. Both of the MT engines we describe in this paper are based on phrase based SMT [2]. The first MT engine uses the decoder based on A* search using stacks for storing the hypothesis in each iteration of decoding

[1] http://www.darpa.mil/IPTO/programs/transtac/transtac.asp

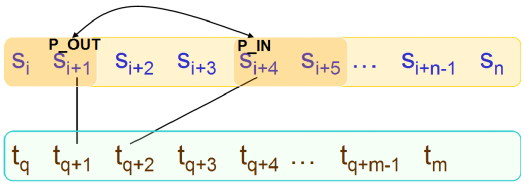

Fig. 2. Lexicalized Reordering Model

step similar to the decoder described in [2]. The reordering model we use for this decoder is based on lexicalized distortion probabilities similar to [5]. Let us call our stack based decoder $D1$ and its reordering model $R1$.

During the decoding of a sentence using $D1$ decoder, it tries to find the optimal path of reordering and translations by using log linear costs associated with the translation options and the reordering distances (distortions). The costs associated with various distortions are computed at the training time using an automatic alignment obtained from an aligner such as GIZA++ [8]. In our model for decoder $D1$ there are two reordering costs which we call IN and OUT. The IN cost describes the probability that the current word jumps certain distance to introduce the first word in a source phrase. Let us describe this with an example in Figure 2. In this example the target word is t_{q+1} aligned with the source word S_{i+1} and the next target word t_{q+2} is aligned with the S_{i+4}. The IN probability describes the cost associated with the source word S_{i+4} (the first word of a phrase $S_{i+4}...S_{i+5}$). It is probability of a jump from the end of previous phrase to the beginning of the current phrase. On the other hand OUT cost is associated with the last word of the phrase, S_{i+1}, from which jump is being made. In order for the decoder to reorder the source sentence such that S_{i+4} is decoded after S_{i+1} skipping two words the OUT cost of S_{i+1} is added with IN cost of S_{i+4} in maximum entropy framework. The estimation for these reordering probabilities based on word level alignment are computed using Equation 1 where the sum over s, t stands for summing over all the alignments generated by all source and target sentence pairs, and $|s|$ represents the length of the source sentence.

$$P_O(d_k|s_i) = \frac{\sum_{s,t}\sum_{j=1}^{|s|}\Delta_k \cdot f(s_i, s_j)}{\sum_{s,t}\sum_{j=1}^{|s|}\sum_{k=-maxD}^{k=maxD}\Delta_k \cdot f(s_i, s_j)} \quad (1)$$

where,

$$f(s_i, s_j) = \begin{cases} 1 & \text{if } s_i = s_j; \\ 0 & \text{if } s_i \neq s_j; \end{cases}$$

and,

$$\Delta_k = \begin{cases} 1 & \text{if } a_{i+1} - a_i = k; \\ 0 & \text{if } a_{i+1} - a_i \neq k; \end{cases}$$

The lexicalized distortion probability is interpolated with a smoothing function $P_S(d_k)$ as shown in Equation 2. The total smoothed probability is used as a log-linear cost in the maximum entropy model. The smoothing weight α is decided empirically.

$$C_O = log(\alpha P_O(d_k|s_i) + (1-\alpha)P_S(d_k)) \quad (2)$$

4.2. Decoder Two (D2): Multiple-Graph Based Decoder

The second decoder $D2$ we experiment with is based on Viterbi search algorithm that finds the best path on a set of FST graphs. The decoder searches on reordering, translation and language model graphs at the same time adding costs across three graphs to find the best path. The reordering graph is an FST built on the fly based on the words of a test sentence such that input for each arc is the source word [9]. The state ID of the reordering FST is binary bit vector that identifies the word positions that have been translated. The bit vector is added with a dot such that the dot locates the last word that had been translated. For example if the words in the example shown in Table 1 were translated in the order of words 1, 2, 4 and 3 we would be getting the following state sequence 1.000, 11.00, 1101. and 111.1 .

Both of the current lexicalized models described above do not use any POS information. The distortions are just based on the words and the alignments. Even though this has shown to be useful for many languages, it is weak for language pairs where the ordering of words differ significantly. The question is how can we exploit the information that verb ordering is different without changing the decoder much and without having to use POS tagger during decoding phase but still fulfill the constraints we described previously. We next present our such method.

5. POS BASED LEXICALIZED REORDERING MODEL

For the sequence of source words in our example shown in Figure 1 we can obtain POS tags using any POS tagger. If the POS sequence is known for our example then we would have known "have" is a verb and is likely to end up in the end of the sentence. We formalize this by assuming that we have a sequence of POS information for the source side sentence **in the training time**. We denote the POS sequence by $P_i...P_n$.

We reiterate the fact that we need POS only during the training time and not during the decoding phase. We integrate POS information in the current training framework using Equation 3. We should note that the current distortion d_k is conditioned not only on the source word identity s_i but also in its POS tag p_i. The term $f((s_i, p_i), s_j)$ equals to 1 only when $s_i = s_j$ & $p_i = max(p_i)$ where $p_i \subset P$, i.e. we increment the count only for the words that have POS tag that match the most frequent POS for that word.

$$P_{O_{POS}}(d_k|s_i, p_i) = \frac{\sum_{s,t} \sum_{j=1}^{|s|} \Delta_k \cdot f((s_i, p_i), s_j)}{\sum_{s,t} \sum_{j=1}^{|s|} \sum_{k=-maxD}^{k=maxD} \Delta_k \cdot f((s_i, p_i) s_j)} \quad (3)$$

where,

$$f((s_i, p_i), s_j) = \begin{cases} 1 & \text{if } s_i = s_j \text{ \& } p_i = max(p_i) \text{ where } p_i \subset P \\ 0 & \text{if } s_i \neq s_j; \end{cases}$$

and,

$$\Delta_k = \begin{cases} 1 & \text{if } a_{i+1} - a_i = k; \\ 0 & \text{if } a_{i+1} - a_i \neq k; \end{cases}$$

Using the equation above we will obtain word and POS based reordering probabilities for all the jump distances for each word. We then interpolate this probability with the baseline model. The interpolation weight λ is determined empirically. The interpolation allows us to smoothen the POS based reordering probabilities as we may not see all of the word, POS and distortion permutations.

$$C_{O_{POS}} = log(\lambda P_O(d_k|s_i) + (1 - \lambda) P_{O_{POS}}(d_k|s_i, p_i)) \quad (4)$$

We convert the probabilities to the log linear cost as shown in 4. The above equation describes OUT distortion probability. The IN POS based lexicalized distortion probability can also be derived similarly. We should note that the final reordering cost is unique to word-distortion pair due to the max operation in Equation 3.

6. EXPERIMENTS AND RESULTS

We did all of our experiments on the subset of Farsi data provided by Transtac. We had 110K parallel English-Farsi sentences. We had a held out development set and test set which were slightly more than 400 sentences. We first built our baseline phrase based translation model for the decoder $D1$. The alignment was performed using GIZA++ from which we extracted phrase pairs and generated the phrase table. The translation probabilities were computed following [2]. The reordering model was built following Equation 1. We tuned the log linear weights using the maximum error rate training. We then tested our models on a held out test set with 4 references.

The baseline model performance is shown in Table 3. We evaluated the performance of our MT systems using BLEU [10]. The development set performance was 29.28 while the test set was 23.49.

	Dev	Test
Baseline Reordering Model	29.28	23.49
POS based Reordering Model	29.72	24.08

Table 3. POS based Lexicalized Reordering Performance for Stack based Decoder $D1$

We then obtained Stanford's statistical POS tagger [11] to tag all of our training data. After obtaining POS sequence for each training sentence we obtained the reordering probabilities using Equation 3. We then interpolated this reordering model with the word-based reordering model using a λ of 0.5. The final reordering model is a table of distortion probabilities for a given word similar to the baseline model except that the reordering probabilities have changed according to POS information. We replaced our baseline reordering model with our new POS based reordering model while keeping all other components the same including the phrase table and the four gram language model. Our updated model were tuned on the same dev and test set. The results with BLEU metric are shown in Table 3. Our new POS based lexicalized reordering model is 0.44 absolute BLEU points better in the dev set and

0.59 absolute BLEU points in test set. We should note that the gain is obtained without any change in the decoding pipeline. We do not use POS tagger to tag test sentences, nor do we integrate POS information in the decoder. When we manually inspected the reordering table, we saw that verbs had higher probability assigned to high order reordering positions compared to the baseline reordering model.

We then experimented with the reordering model for our second decoder $D2$. We built the baseline model following [9]. The baseline performance is shown in Table 4. We obtained 29.27 BLEU score for development set and 26.22 BLEU score for the test set. The baseline test set performance for $D2$ is better than the baseline performance for our $D1$ decoder. This is probably due to the fact that the reordering graph generated on the fly for the baseline model of $D2$ searches over all possible reordering for the given window size while the $D1$ decoder prunes the search space by pruning the hypothesis stack.

We went through the same process of using POS tags for all the training data and computed the distortion probability based on word and POS using Equation 3 which was then interpolated with baseline model to build the final lexicalized reordering model for the decoder $D2$. The POS based model's performance is shown in Table 4. We can note that the new model's performance is not much better than the baseline model. There are a couple possible reasons for less gain for decoder $D2$. First, as we explained above, $D2$ allows more flexible reordering than $D1$, without pruning the reordering permutations in each step of decoding. Second, we used the reordering window size of 5 which meant that decoder could not move the words further than 5 word positions even though the reordering table could give probability mass to positions beyond that. 5 is the maximum window size we can use for evaluation without seeing significant degradation in latency of S2S system. Hence we had to stick with the window size of 5 possibly lowering the performance gain that we could have gotten with a higher window size.

	Dev	Test
Baseline Reordering Model	29.27	26.22
POS Based Reordering Model	29.31	26.23

Table 4. POS based Lexicalized Reordering Performance for FST based Decoder $D2$

7. CONCLUSION

We presented a technique to integrate POS information to MT engines without adding complexity to the decoder and without the need of POS tagger in the runtime to tag test sentences. Even though we can potentially get more gains by adding POS information within the decoder itself for language pairs where the ordering differs greatly we would also need significant changes to the overall MT pipeline. This may not be ideal for developing MT systems within the constraints of latency, time and memory needed for S2S system such as the ones in Transtac program. We presented our empirical results for two different types of decoding algorithms based on Viterbi search using FST framework and A* search using stacks respectively. We showed that we can rapidly integrate POS information for re-estimating lexicalized reordering distance probabilities and interpolating them with a baseline model to get improvements without adding complexity.

8. ACKNOWLEDGMENT

This work is in part supported by the US DARPA under the TransTac program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. We would also like to acknowledge Yongang Deng for helpful discussions.

9. REFERENCES

[1] Peter E Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer, "The mathematics of statistical machine translation: parameter estimation," *Computational Linguistics*, vol. 19, pp. 263–311, 1993.

[2] Philipp Koehn Franz, Franz Josef Och, and Daniel Marcu, "Statistical phrase-based translation," in *Proceedings of HLT/NAACL Conference*, 2003.

[3] David Chiang, "A hierarchical phrase-based model for statistical machine translation," in *ACL*, 2005.

[4] Christop Tillman, "A block orientation model for statistical machine translation," in *Proceedings of JLT*, 2004.

[5] Yaser Al-Onaizan and Kishore Papineni, "Distortion models for statistical machine translation," in *Proceedings of ACL*, 2006.

[6] Kenji Yamada and Kevin Knight, "A syntax-based statistical translation model," in *Proceedings of ACL*, Toulouse, France, July 2001, pp. 523–530, ACL.

[7] Michael Collins, Philipp Koehn, and Ivona Kucerova, "Clause restructuring for statistical machine translation," in *Proceedings of ACL*, 2005.

[8] Franz Josef Och and Hermann Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[9] Bowen Zhou, Rong Zhang, and Yuqing Gao, "Lexicalized reordering in multiple-graph based statistical machine translation," in *Proceeding of ICASSP*, 2008.

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu, "Bleu: A method for automatic evaluation of machine translation," in *ACL*, 2002.

[11] Kristina Toutanvoa and Christopher D. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *proceedings of SIGDAT*, Hong Kong, China, 2000.