Dealing with deletion errors in MT

Jacob Devlin

June 9, 2008

1 Introduction

Content word deletion has widely been identified as a common error in statistical machine translation systems. We refer to a translation error as "content word deletion" when a content word appears in the reference translation of some sentence, but is absent from the system output translation of that sentence. For our purposes, the term "content word" does not have a precise definition, but represents any word that we judge to carry semantic meaning in the sentence. We hypothesize that a major source of content word deletion is the use of translation rules that contain unaligned content words on the source side. The following is a rule of this type:

Source: 指出 ,
Target: ,

In this rule, the Chinese content word 指出, which translates to "pointed out" in English, aligns to nothing on the target side of the rule. Put simply, this rule says that if we see the Chinese word for "pointed out" followed by a comma, we can translate these two words to just a comma in English. The use of the rule above in our translation system was the direct cause of the content word deletion error seen here:

Source:	伦敦 每日 快报 指出 , 两 台 记载 黛安娜 ...
System Output:	the london daily express , two portable computer ...
Reference:	the london daily express pointed out that two laptop computer ...

Preliminary experiments show that simply creating a universal penalty for all rules with unaligned source words does not improve the quality of our translation system. In addition, we intuitively believe that more refined penalization methods are necessary, because there are many Chinese function words such as 的 which have no direct meaning in English, and even the Chinese word discussed above 指出 can be left out of the translation in some contexts. In the following example from the training data, 指出 is unaligned, but the English translation is completely fluent:

Source: 但 美国 舆论 指出 , 扩军 将 大大 增加 美国 的 国防 开支
Reference: but according to us public opinion , expansion of the army will greatly increase us defense expenditure .

Thus, the goal of this project is *context dependent* penalization of rules with unaligned source words. Here, "context dependent" means that the penalty (i.e. feature score) is a function of the unaligned word itself, the other words surrounding the sentence, the structure of the rule, and so on.

In section 2, we will give a brief overview of our current, state of the art translation system. This section will also explain exactly what it means for a word to be unaligned, and how unaligned words end up in our translation rules. In section 3 we will provide details of the various features that we implemented in our attempt to reduce content word deletion errors. In section 4, we will present our results.

2 Overview of the statistical machine translation system

We use state of the art hierarchcial MT system based off of David Chiang's Hiero [2]. The MT system can be divded into three major steps: Alignment of parallel training data, rule creation/generalization, and decoding. An overview of each step is provided below.

2.1 Alignment of parallel training data

Given a source/target sentence pair (s_1^J, t_1^I), we denote an alignment between these two sentence as a_1^J, where the j^{th} source word is aligned to (i.e. translates to) the ${a_j}^{th}$ target word. The special alignment $a_j = 0$ means that the source word at index j does not align to any target words. In a more intuitive definition, $a_j = 0$ says that meaning of word j is not semantically encoded in any words in the target side of the training sentence. We use GIZA++ with IBM models 1-4 [1] and the HMM model [7] to align the parallel training corpus. Given a set of K sentence pairs $\{(\mathbf{s_k}, \mathbf{t_k}), k = 1...K\}$, the goal is to find the set of hidden parameters θ that maximizes the likelihood of the training corpus [4]:

$$\hat{\theta} = \arg\max_{\theta} \prod_{k=1}^{K} \sum_{\mathbf{a}} p_\theta(\mathbf{s_k}, \mathbf{a}|\mathbf{t_k}) \tag{1}$$

Where $p_\theta(\mathbf{s_k}, \mathbf{a}|\mathbf{t_k})$ represents the probability that $\mathbf{s_k}$ maps to $\mathbf{t_k}$ using the alignment $\mathbf{a}$, under the hidden parameters θ. We determine the hidden parameters $\hat{\theta}$ using the EM algorithm. Once this has been determined, we can find the highest probability alignment $\hat{a}_1^J$ between each sentence pair s_1^J and t_1^I, which is known as the **Viterbi alignment**:

$$\hat{a}_1^J = \arg\max_{a_1^J} p_{\hat{\theta}}(s_1^J, a_1^J|t_1^I)$$

The implications of this alignment model are as follows:

- Some words in the source language may be unaligned
- Some words in the target language may be unaligned
- Multiple words in the source language can be aligned to the same word in the target language
- A word in the source language *cannot* be aligned to multiple words in the target language

In order to get around these limitations, we perform a "backward" alignment b_1^I from the target language to the source language, which is done by running GIZA++ again with the the source language and target language switched. We then combine the alignment vectors a_1^J and b_1^I to create an $I \times J$ alignment matrix A. If we let $A_1 = \{(a_j, j)|a_j > 0\}$ and $A_2 = \{(i, b_i)|b_i > 0\}$, then we let our final alignment matrix $A = \mathit{combine}(A_1, A_2)$. The *combine*() function is described in Och et al. (2004) as the "refined method" for combining forward and backward alignment. This refined method can be thought of as being "between" the set intersection and set union, i.e. $(A_1 \cap A_2) \subseteq \mathit{combine}(A_1, A_2) \subseteq (A_1 \cup A_2)$.

The resulting alignment A is a many-to-many mapping between the source and target sentences, where any number of words on both the source and target side may be unaligned. The aligned parallel corpus, represented as a set of K triples (s_k, t_k, A_k), is directly used as the input to the rule extraction process.

2.2 Extraction of translation rules

We use a hierarchical rule extraction process as described in Chiang et al. (2005)[2]. As a first step, all phrase translations are extracted from each training triple $(\mathbf{s_k}, \mathbf{t_k}, \mathbf{A_k})$. We extract each possible phrase translation $\langle \hat{s}, \hat{t} \rangle$ such that no words in $\hat{s}$ are aligned to any other target words (other than the ones in $\hat{t}$), and no words in $\hat{t}$ are aligned to any other source source words (other than the ones in $\hat{t}$). For the purpose of this paper, it is important to note is that unaligned words may appear on the edges or in the center of these rules.

We convert the phrase rules to hierarchical rules by first converting each phrase pair $\langle \hat{s}, \hat{t} \rangle$ to a CFG rule $X \rightarrow \langle \hat{s}, \hat{t} \rangle$, and then subtracting each phrase pair $\langle \hat{s}, \hat{t} \rangle$ from each rule in the form $X \rightarrow \langle \gamma_1 \hat{s} \gamma_2, \alpha_1 \hat{t} \alpha_2 \rangle$ to create the hierarchical rule $X \rightarrow \langle \gamma_1 X \gamma_2, \alpha_1 X \alpha_2 \rangle$. In order to reduce the number of possible rules that can be created, we apply restrictions to the size of phrases that can be extracted and subtracted, and we also apply filtering based on the current test set. Joint and marginal counts are then summed over all the rules.

2.3 Decoding

For decoding, we use a hierarchical model that closely follows Chiang's Hiero, as described in Chiang (2005). In this model, we create a shared forest of weighted translation rules

for the sentence being decoded. We use a log linear model to score each translation rule:

$$\begin{aligned} w(X \rightarrow \langle \gamma, \alpha \rangle) &= \prod_i \phi_i(X \rightarrow \langle \gamma, \alpha \rangle)^{\delta_i} \\ \log(w(X \rightarrow \langle \gamma, \alpha \rangle)) &= \sum_i \delta_i \phi_i(X \rightarrow \langle \gamma, \alpha \rangle) \end{aligned}$$

Since hierarchcial rules form a CFG, the test sentence is parsed on the source side (using a chart parser similar to CKY), creating a shared forest of target derivations. The score of a derivation (i.e. a full sentence) is thus the sum of the scores of each rule used in that derivation. Each rule can have an arbitrary number of features, and the feature weights are generally determined automatically through optimization towards an evaluation metric such as BLEU [5]. Some basic features include:

- Language model score
- $P(\gamma|\alpha)$ [3]
- $P(\alpha|\gamma)$
- Lexical probability
- Word penalty, i.e. a feature equal to the number of words in α

We can easily extend the decoding model by adding new features that are functions of the rule $X \rightarrow (\gamma, \alpha)$, the sentence being decoded (including parsing), outside decoding statistics, and so on. The next section describes the features that we added in order to reduce content word deletion errors.

3 New decoding features

We created and tested a number of features with the aim of reducing content word deletion errors. These were implemented as a decoding feature within our log-linear model, and thus the weights for each feature were optimized automatically towards BLEU. The specifics of each feature are described below. Multiple features were also tested at the same time, where it made sense.

Note: The following refers to the feature score some translation rule R with unaligned source word W. For example, if R is the rule given in section 1 above, then W would be the word 指出. Some rules may have more than one unaligned source word, in which case the feature scores for each W were calculated independently, and then summed to provide the total feature score for R.

1. **Whether or not W appeared on a manually created function word list.** A Chinese speaker created a list of roughly one hundred Chinese function words, and R was not penalized if W was included in this list. The idea behind this was that function words usually require no direct translation, so we shouldn't penalize rules that have unaligned function words.

2. **Whether W appeared on the edge or in the middle of the source side of R.**
   W is considered in the middle of R if there are aligned words (terminals or non-terminals) to its left and right. Otherwise, W is considered on the edge of R. For example, in (1), 指出 is in the middle of the rule, since it has an aligned ',' to its right and an aligned non-terminal 'X' to its left. The idea behind this feature was that it is more 'dangerous' to include rules with unaligned words on the edge of the source phrase during rule extraction, and so we would want to penalize these rules more harshly.

3. **The percent of time that W was unaligned in the training data.**
   For each word s in the training data, we calculated the counts $C(s)$ and $C_{unalign}(s)$, where $C(s)$ is the number of times s was seen total and $C_unalign(s)$ is the number of times s was seen unaligned. The percentage value is simply $C_{unalign}(s)/C(s)$. We tried this value directly and with simple smoothing.

4. **The percent of time that W's part-of-speech tag was unaligned in the training data.**
   Since the parallel corpus was tagged, we iterated through and calculated $C(t)$ and $C_{unalign}(t)$, for every Chinese part-of-speech tag t. "W's part-of-speech tag" is the tag that W received in the context of the test sentence, and is not related to rule R or the training data.

5. **The percent of time that W + part-of-speech was unaligned in the training data.**
   The counts were calculated in the same way as above, except that every word and part-of-speech combination was counted separately. So $C(\Upsilon, Verb)$ would be the total number of times that 指出 was seen tagged as a Verb, and $C_{unalign}(\Upsilon, Verb)$ would be the total number of times that 指出 was seen tagged as a Verb, and wasn't aligned to anything on the English side.

6. **Discrete "bins" of the above percentages.**
   If the percent of time that W (or W's part of speech, or W+POS) is unaligned in the training data is denoted as p, then the penalty score for rule R is $f(p)$, where f is some function. In most of our experiments, we used $f(p) = p$, e.g. if W was unaligned 25% of the time in the training data, the penalty would simply be 0.25. In our current translation model, we assign an optimized weight z to each feature, so that the actual penalty on R is $f(p)z$. Note that since $p < 1.0$ (assuming W is unaligned even once), if f(p) = p then this feature will always penalize the rule, even if W is unaligned 95% of the time, because $0.95x < 1$ when $x > 0$. Intuitively, we would think that if W was unaligned 95% of the time, then we shouldn't penalize the rule at all, and we might even want to reward the rule. However, coming up with a good function for f is difficult, and it is not something that can be done through automatic optimization. Instead we can create n different features, and then put each penalty score into a feature "bin," so we can automatically optimize a good approximation of f. For example, say we have two features bins corresponding to $0.0 \leq p < 0.7$ and $0.7 \leq p \leq 1.0$ and two weights z_1 and z_2 corresponding to the

two bins. If $z_1 > 0$ and $z_2 < 0$, then $p < 0.7$ would penalize R, while a $p \geq 0.7$ would reward R. Or more realistically it could end up that $z_1 > z_2 > 0$, and thus we penalize lower p values more harshly. Note that these weights z_i are found using automatic optimization, but the number of bins and the ranges for each bin cannot be optimized automatically.

7. **The translation entropy of *W*.**
   This was a simple entropy calculation, where the probability distribution was the translation probabilities of all the translation rules that had W as the rule's source phrase.

8. ***W*hether or not there are unaligned words in the target side of R.**
   The penalty scores were put into separate features depending on whether or not there was at least one unaligned word on the target side of R. The idea was that an unaligned word on the target side of R might be a translation of W, so we would want to penalize R less. However, our tests showed that rules of this type were extremely rare.

4 Results

The results are presented in Table 1. The numbers in the "experiment" column refer to the feature conditions listed above. Unfortunately, none of the conditions tested showed a consistent gain in TER and BLEU scores. Manual analysis of the results show that many deletion errors were fixed, but a number of "insertion errors" were introduced, likely offsetting the gain.

Table 1: BLEU/TER scores of test conditions on NIST test set

Experiment	BLEU	TER
Baseline	40.31	52.11
(1)+(2)	40.23	52.14
(3)+(2)	40.05	52.09
(4)	39.97	52.25
(5)	40.86	52.25
(5) + (6)	39.95	52.04
(7)	40.05	52.05
(8)	40.24	51.93

The following example demonstrates how the deletion error shown in the introduction was fixed from the use of feature (5):

Baseline: the london daily express , two portable computer ...
Feature (5): london daily express **said** , two laptop computer ..
Reference: london daily express pointed out that two laptop computers ...

However, the use of feature (5) also caused an extraneous insertion in the following example. The word "bosnia" does not appear in the baseline output or reference:

Baseline:	... austria , bulgaria , croatia , germany , greece , italy , romania , slovakia , ukraine and russia and europe ...
Feature (5):	... austria , bulgaria , croatia , germany , greece , italy , romania , slovenia , **bosnia** , ukraine and russia's territory in europe ...
Reference:	... austria , bulgaria , croatia , germany , greece , italy , romania , slovenia , ukraine , and the european territory of russia ....

It is important to note that in the example where the deletion error was fixed, the sentence not receive a better TER or BLEU score because "said" does not exactly match "pointed out". This is a known and widely discussed weakness in the TER and BLEU evaluation metrics, and the general conclusion is that if some new feature really does cause the system to produce better translations, then this will be reflected by BLEU and TER over a large number of sentences. In other words, we could reason that if a feature actually fixes more deletion errors than it does create new insertion errors, then the BLEU/TER scores over the whole test set should be better. However, when manually comparing the baseline output to the output of a test condition (feature (5)), there appear to be many more cases of fixed deletion errors than new insertion errors. Thus, we believe that there may be an increase in the quality of translation output from the use of these features, even though it is not reflected in the TER and BLEU scores. In order to determine if this is the case, an HTER evaluation would be needed on a significant number of sentences. HTER involves human annotation of the system output and references, and unlike TER or BLEU, HTER factors in semantic equivalence between words in the system output and words in the reference translation [6].

Despite the lack of conclusive positive results, we still feel that content word deletion errors are a major problem in statistical machine translation, and that unaligned source words are a major cause of these errors. We will continue to work to reduce these errors through the use of novel decoding features and more refined evaluation methods.

References

[1] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematic of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1994.

[2] David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. The hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 779–786, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[3] Philipp Koehn, Franz Josef Och, and Daniel. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133, 2003.

[4] Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, pages 417–449, 2004.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[6] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Association for Machine Translation in the Americas*, 2006.

[7] Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836–841, Morristown, NJ, USA, 1996. Association for Computational Linguistics.