Microsoft Research Treelet Translation System: IWSLT Evaluation

Arul Menezes and Chris Quirk

Microsoft Research
One Microsoft Way
Redmond, WA 98052
{arulm,chrisq}@microsoft.com

Abstract

The Microsoft Research translation system is a syntactically informed phrasal SMT system that uses a phrase translation model based on dependency treelets and a global reordering model based on the source dependency tree. These models are combined with several other knowledge sources in a log-linear manner. The weights of the individual components in the log-linear model are set by an automatic parameter-tuning method. We give a brief overview of the components of the system and discuss its performance at IWSLT in two tracks: Japanese to English (supplied data and tools), and English to Chinese (supplied data and tools).

1. Introduction

The dependency treelet translation system developed at MSR is a statistical MT system that takes advantage of linguistic tools, namely a source language dependency parser, as well as a word alignment component. [1]

To train a translation system, we require a sentence-aligned parallel corpus. First the source side is parsed to obtain dependency trees. Next the corpus is word-aligned, and the source dependencies are projected onto the target sentences using the word alignments. From the aligned dependency corpus we extract all treelet translation pairs, and train an order model and a bi-lexical dependency model.

To translate, we parse the input sentence, and employ a decoder to find a combination and ordering of treelet translation pairs that cover the source tree and are optimal according to a set of models. In a now-common generalization of the classic noisy-channel framework, we use a log-linear combination of models [2], as in below:

$$translation(\mathbf{S},\mathbf{F},\mathbf{\Lambda}) = \arg\max_{\mathbf{T}}\left\{\sum_{f\in\mathbf{F}} \lambda_f \, f(\mathbf{S},\mathbf{T})\right\} \quad (3)$$

Such an approach toward translation scoring has proven very effective in practice, as it allows a translation system to incorporate information from a variety of probabilistic or non-probabilistic sources. The weights $\mathbf{\Lambda} = \{ \lambda_f \}$ are selected by discriminatively training against held out data.

2. System Details

A brief word on notation: *s* and *t* represent source and target lexical nodes; **S** and **T** represent source and target trees; **s** and **t** represent source and target treelets (connected subgraphs of the dependency tree). Where the intent is clear, we will disregard the structure of these elements and consider these structures to be sets of lexical items: the expression $\forall t \in \mathbf{T}$ refers to all the lexical items in the target language tree **T**. Similarly, $|\mathbf{T}|$ refers to the count of lexical items in **T**. We use subscripts to indicate selected words: $\mathbf{T}_n$ represents the n^{th} lexical item in an in-order traversal of **T**.

2.1. Training

We use the broad coverage dependency parser NLPWIN [3] to obtain source language dependency trees, and we use GIZA++ [4] to produce word alignments. The GIZA++ training regimen and parameters are tuned to optimize BLEU [5] scores on held-out data. Using the word alignments, we follow a set of dependency tree projection heuristics [1] to construct target dependency trees, producing a word-aligned parallel dependency tree corpus. Treelet translation pairs are extracted by enumerating all source treelets (to a maximum size) aligned to a target treelet.

2.2. Decoding

We use a tree-based decoder, inspired by dynamic programming. It searches for an approximation of the n-best translations of each subtree of the input dependency tree. Translation candidates are composed from treelet translation pairs extracted from the training corpus. This process is described in more detail in [1].

2.3. Models

2.3.1. Channel models

We employ several channel models: a direct maximum likelihood estimate of the probability of target given source, as well as an estimate of source given target and target given source using the word-based IBM Model 1 [6]. For MLE, we use absolute discounting to smooth the probabilities:

$$\mathrm{P}_{MLE}(\mathbf{t} \mid \mathbf{s}) = \frac{c(\mathbf{s},\mathbf{t}) - \lambda}{c(\mathbf{s},*)} \quad (4)$$

Here, *c* represents the count of instances of the treelet pair ⟨**s, t**⟩ in the training corpus, and λ is determined empirically.

For Model 1 probabilities we compute the sum over all possible alignments of the treelet without normalizing for length. The calculation of source given target is presented below; target given source is calculated symmetrically.

$$\mathrm{P}_{M1}(\mathbf{t} \mid \mathbf{s}) = \prod_{t\in\mathbf{t}} \sum_{s\in\mathbf{s}} \mathrm{P}(t \mid s) \quad (5)$$

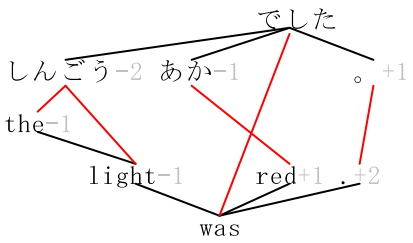

Figure 1: Aligned dependency tree pair, annotated with head-relative positions

2.3.2. *Target language models*

We use both a surface level trigram language model and a dependency-based bigram language model [7], similar to the bilexical dependency modes used in some English Treebank parsers (e.g. [8]).

$$\begin{aligned} P_{surf}(\mathbf{T}) &= \prod_{i=1}^{|\mathbf{T}|} P_{trisurf}(\mathbf{T}_i \mid \mathbf{T}_{i-2}, \mathbf{T}_{i-1}) \\ P_{dep}(\mathbf{T}) &= \prod_{i=1}^{|\mathbf{T}|} P_{bidep}(\mathbf{T}_i \mid parent(\mathbf{T}_i)) \end{aligned} \quad (6)$$

$P_{trisurf}$ is a Kneser-Ney smoothed trigram language model trained on the target side of the training corpus, and P_{bilex} is a Kneser-Ney smoothed bigram language model trained on target language dependencies extracted from the aligned parallel dependency tree corpus.

2.3.3. *Order model*

The order model attempts to assign a probability to the position (*pos*) of each target node relative to its head based on information in both the source and target trees:

$$P_{order}(order(\mathbf{T}) \mid \mathbf{S}, \mathbf{T}) = \prod_{t \in \mathbf{T}} P(pos(t, parent(t)) \mid \mathbf{S}, \mathbf{T}) \quad (7)$$

Here, position is modeled in terms of closeness to head in the dependency tree. The closest pre-modifier of given head has position -1; the closest post-modifier has a position 1. Figure 1 shows an example dependency tree pair annotated with head-relative positions.

We use a small set of features reflecting local information in the dependency tree to model P(pos(*t*,parent(*t*)) | **S**, **T**):

- Lexical items of *t* and *parent*(*t*).
- Lexical items of the source nodes aligned to *t* and head(*t*).
- Part-of-speech ("cat") of the source nodes aligned to the head and modifier.
- Head-relative position of the source node aligned to the source modifier.

These features along with the target feature are gathered from the word-aligned parallel dependency tree corpus and used to train a decision tree. [9]

2.3.4. *Other models*

In addition to these basic models, we also incorporate a variety of other information to gather information about the translation process.

- **Treelet count**. This feature is a count of treelets used to construct the candidate. It acts as a bias toward translations that use a smaller number of treelets; hence toward larger sized treelets incorporating more context.
- **Word count.** We also include a count of the words in the target sentence. This feature helps to offset the bias of the target language model toward shorter sentences.
- **Whole sentence Model 1 scores.** We provide the system with both the probability of the whole source sentence given the whole target sentence and vice versa, as described in [10]:

$$P(\mathbf{S} \mid \mathbf{T}) = \frac{\varepsilon}{(|\mathbf{T}|+1)^{|\mathbf{S}|}} \prod_{s \in \mathbf{S}} \sum_{t \in \mathbf{T}} P(s \mid t) \quad (8)$$

- **Deletion penalty.** As in [11], we approximate the number of deleted words using Model 1 probabilities using the following formula, where *d* is an empirically determined threshold:

$$D(\mathbf{S}, \mathbf{T}) = \left|\{s \in \mathbf{S} \mid \forall t \in \mathbf{T}. P(t \mid s) < d\}\right| \quad (9)$$

- **Insertion penalty.** An approximation of the number of insertions can be counted in the same manner:

$$I(\mathbf{S}, \mathbf{T}) = \left|\{t \in \mathbf{T} \mid \forall s \in \mathbf{S}. P(s \mid t) < i\}\right| \quad (10)$$

2.4. Small corpus optimizations

2.4.1. *Model 1 fallback translations*

Due to the small size of the training corpus, we found that misalignments often prevented us from finding treelet translation pairs for rare tokens, or would lead to very poor translations. All too often, too much source or target context would be aligned with the rare word, thus leading to either untranslated source words in the output, or spurious additional words in the translation. Absolute discounting of the MLE channel model helps to solve the latter issue, but exacerbates the former. To work around this situation, at runtime we construct single-word treelet translation pairs for each input node from the top few entries in the Model 1 translation table.

2.4.2. *NULL translations*

We also create treelets that allow words to be translated as the empty token, i.e. to be deleted. These NULL translations are assigned a default MLE probability.

2.5. Corpus specific issues

2.5.1. *Data cleanup*

We observed in many cases that individual training pairs consisted of several concatenated sentences in one or both languages. Since our system parses the source language, we prefer to process sentences individually; hence we broke these multi-sentence utterances into individual sentences and used simple positional heuristics to re-align the sentences.

Japanese to English: Our Japanese parser prefers to do its own word-breaking, so we removed all spaces from the word-

Table 1: Summary of results under a variety of configurations.
Final submission results are shown in **boldface**. Best results in each category are shown in *italics*

System	BLEU-4 / NIST			
	2003 Devset	2004 Devset	Final	Final w/ Verbatim
Japanese to English (best NIST parameters)				
Baseline	48.8 / 8.7	48.4 / 8.4	39.3 / 7.4	39.6 / 7.4
Learned normalization	49.6 / 9.1	48.9 / 9.1	40.3 / 8.2	*41.1 / 8.2*
Hiragana normalization	49.7 / 9.2	49.3 / 9.2	40.1 / 8.0	**40.6 / 8.0**
Japanese to English (best BLEU parameters)				
Baseline	49.5 / 7.1	48.1 / 7.3	39.6 / 6.7	40.1 / 6.8
Learned normalization	49.0 / 8.6	49.2 / 8.6	42.1 / 7.7	*42.9 / 7.8*
Hiragana normalization	51.0 / 8.8	50.3 / 8.5	40.8 / 7.4	41.2 / 7.4
English to Chinese				
	14.6 / 3.8	17.3 / 4.4	*22.0 / 6.5*	**20.6 / 5.2**

broken Japanese text. We also found it advantageous to remove (*after* parsing) certain lexical items, such as the topic marker ("は"), the politeness prefix "御", and the modal expression "のです". And finally we normalize the sequence of a question particle and a period ("か。") into a single question mark character ("?").

2.5.2. Japanese character set normalization

Japanese is commonly written in a combination of three distinct writing systems: kanji (ideographic), hiragana (syllabic), and katakana (syllabic). In most situations, there is a canonical spelling of a lexical item; however certain items admit multiple spellings (e.g. ください/下さい or わかりません/分かりません). Such ambiguity exacerbates the data sparsity problem already evident in the small training corpus. In addition the distribution of hiragana to kanji spellings in the final testing data is noticeably different than that of the training set and development sets. We establish a baseline and propose two corrections for this situation.

Baseline: *No normalization*. In this version of the system, no character set normalization is performed.

Method 1: *Hiragana normalization*. After parsing the input, we look up the dictionary headword of each node in the tree, and, if present in the dictionary, replace that lexical item with its lemmatized Hiragana form. The mapping is lossy in that distinct words with different kanji representations but identical hiragana representations are conflated. Also morphological endings on words are lost.

Method 2: *Learned normalization*. We observed that between the old (unbroken) and new (word-broken) distributions of the training data a significant number of character set changes had been made to the data, in addition to the word-breaking. We used these differences to automatically acquire a character-set normalization table that was then applied to test and training data.

2.5.3. Verbatim translations

This corpus contains large amount of repetitive phrases; to a large extent, this comes with the domain. Basic travel expressions are commonly short, simple phrases like "all right" or "thanks". To minimize errors on such stock phrases, we introduce a verbatim component, a type of translation memory. If the input sentence matches the source segment of a training pair, we return the target side of the training pair as the translation.

2.6. Parameter training and tuning

There are several types of tunable parameters. First we have Λ, a 12 dimensional weight vector: one real-valued weight for each feature function. This weight vector is determined by maximization of the BLEU score using n-best lists [12].

However, there are a variety of other parameters that cannot be tuned via n-best lists; instead these are optimized by grid search on BLEU score.

- **Maximum treelet size**: keep treelet translation pairs up to *s* nodes (for both JE and EC, the optimal value was 9).
- **Treelet translation pair pruning cutoff**: explore only the top *k* treelet translation pairs per input node (JE: 9; EC: 6).
- **Decoder beam width**: keep only *n* best translated subtrees per input subtree (JE: 15; EC: 12).
- **Exhaustive ordering threshold**: fall back to greedy ordering when the count of children whose order is unspecified exceeds this limit; see [1] for details (JE: 7; EC: 6).
- **MLE channel model discount** (JE and EC: 0.7).
- **Deletion and insertion penalty cutoffs** (JE and EC: 0.1).
- **Default NULL translation probability** (JE: 0; EC: 0.1).

3. Discussion

We present results in two tracks: Japanese to English (supplied data and tools), and English to Chinese (supplied data and tools). Table 1 summarizes the results in these categories. Several trends are worthy of note.

First, it seems that kanji/hiragana normalization is an important component of Japanese to English translation. Unfortunately the more promising of the two methods on the development sets turned out to be less effective on the final training set.

Secondly, we find surprisingly mixed results from using verbatim translations. While their addition has a small but respectable impact on Japanese to English translation (which uses 16 reference translations), the impact is strongly negative on English to Chinese translation (which uses only 1 reference). Most likely this is because, even though the

translations obtained in this manner are almost definitely good translations, they may not match the single reference.

Finally, we note the tradeoff between optimal BLEU scores and optimal NIST scores in Japanese to English translation. While the NIST score has a harsh brevity penalty, BLEU is much more tolerant of very short translations, hence some systems may produce misleadingly large BLEU scores by producing very short translations. Considering both metrics simultaneously helps to identify when this situation is occurring. In this evaluation we observe this to be a major issue across all the participants. For example, in the Japanese to English Supplied Data track, ordering systems by NIST score vs. BLEU score produces a significantly different ranking. The system that ranks first by BLEU drops to fifth when ranked by NIST, whereas the systems ranking first and second by NIST drop to third and sixth place by BLEU.

4. Acknowledgements

We would like to thank Hisami Suzuki and Chris Brockett for suggestions and improvements to the Japanese analysis components; Kevin Duh for analysis of Chinese output; Robert C. Moore for suggestions on smoothing; and the IWSLT organizers, especially Chiori Hori and Mattias Eck, for their feedback and assistance throughout the evaluation.

5. References

[1] Quirk, C., Menezes, A., and Cherry, C., "Dependency Tree Translation: Syntactically Informed Phrasal SMT", *Proceedings of ACL 2005*, Ann Arbor, MI, USA, 2005.

[2] Och, F. J., and Ney, H., "Discriminative Training and Maximum Entropy Models for Statistical Machine Translation", *Proceedings of ACL 2002*, Philadelphia, PA, USA, 2002.

[3] Heidorn, G., “Intelligent writing assistance”, in Dale et al. *Handbook of Natural Language Processing*, Marcel Dekker, 2000.

[4] Och, F. J., and Ney H., "A Systematic Comparison of Various Statistical Alignment Models", *Computational Linguistics*, 29(1):19-51, March 2003.

[5] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J., "BLEU: a method for automatic evaluation of machine translation", *Proceedings of ACL 2002*, Philadelphia, PA, USA, 2002.

[6] Brown, P. F., Della Pietra, S., Della Pietra, V. J., and Mercer, R. L., "The Mathematics of Statistical Machine Translation: Parameter Estimation", *Computational Linguistics* 19(2): 263-311, 1994.

[7] Aue, A., Menezes, A., Moore, R., Quirk, C., and Ringger, E., "Statistical Machine Translation Using Labeled Semantic Dependency Graphs." *Proceedings of TMI 2004*, Baltimore, MD, USA, 2004.

[8] Collins, M., "Three generative, lexicalised models for statistical parsing", *Proceedings of ACL 1997*, Madrid, Spain, 1997.

[9] Chickering, D.M., "The WinMine Toolkit", Microsoft Research Technical Report MSR-TR-2002-103, Redmond, WA, USA, 2002.

[10] Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., and Radev, D., "A Smorgasbord of Features for Statistical Machine Translation". *Proceedings of HLT/NAACL 2004*, Boston, MA, USA, 2004.

[11] Bender, O., Zens, R., Matsuov, E. and Ney, H., "Alignment Templates: the RWTH SMT System". *IWSLT Workshop at INTERSPEECH 2004*, Jeju Island, Korea, 2004.

[12] Och, F. J., "Minimum Error Rate Training for Statistical Machine Translation", *Proceedings of ACL 2003*, Sapporo, Japan, 2003.