Source-Side Left-to-Right or Target-Side Left-to-Right? An Empirical Comparison of Two Phrase-Based Decoding Algorithms

Yin-Wen Chang
Google Research, New York
yinwen@google.com

Michael Collins*
Google Research, New York
mjcollins@google.com

Abstract

This paper describes an empirical study of the phrase-based decoding algorithm proposed by Chang and Collins (2017). The algorithm produces a translation by processing the source-language sentence in strictly left-to-right order, differing from commonly used approaches that build the target-language sentence in left-to-right order. Our results show that the new algorithm is competitive with Moses (Koehn et al., 2007) in terms of both speed and BLEU scores.

1 Introduction

Phrase-based models (Koehn et al., 2003; Och and Ney, 2004) have until recently been a state-of-the-art method for statistical machine translation, and Moses (Koehn et al., 2007) is one of the most used phrase-based translation systems. Moses uses a beam search decoder based on a dynamic programming algorithm that constructs the target-language sentence from left to right (Koehn et al., 2003). Neural machine translation systems (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), which have given impressive improvements over phrase-based systems, also typically use models and decoders that construct the target-language string in strictly left-to-right order.

Recently, Chang and Collins (2017) proposed a phrase-based decoding algorithm that processes the *source-language* string in strictly left-to-right order. Reordering is implemented by maintaining multiple sub-strings in the target-language, with phrases being used to extend these sub-strings by various operations (see Section 2 for a full description). With a fixed distortion limit on reordering, the time complexity of the algorithm is linear in terms of sentence length, and is polynomial time in other factors.

Chang and Collins (2017) present the algorithm and give a proof of its time complexity, but do not describe experiments, leaving an open question of whether the algorithm is useful in practice. This paper complements the original paper by studying the algorithm empirically. In addition to an exact dynamic programming implementation, we study the use of beam search with the algorithm, and another pruning method that restricts the maximum number of target-language strings maintained at any point. The experiments show that the algorithm is competitive with Moses in terms of both speed and translation quality (BLEU score).

The new decoding algorithm is of interest for a few reasons. While the experiments in this paper are with phrase-based translation systems, the method could potentially be extended to neural translation, for example with an attention-based model that is in some sense monotonic (left-to-right). The decoder may be relevant to work on simultaneous translation (He et al., 2016). The ideas may be applicable to string-to-string transduction problems other than machine translation.

2 A Sketch of the Decoding Algorithm of Chang and Collins (2017)

This section gives a sketch of the decoding algorithm of Chang and Collins (2017). We first define the phrase-based decoding problem, and then describe the algorithm.

2.1 The Phrase-based Decoding Problem

Throughout this paper we will consider the following decoding problem. Given a source sentence $x_1 \dots x_n$ for $n \geq 1$, a phrase $p = (s, t, e)$ specifies a possible translation from $x_s \dots x_t$ to a string

* On leave from Columbia University.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1496–1500
Copenhagen, Denmark, September 7–11, 2017.

Sub-derivation	State
$\langle\langle(\mathbf{1},\mathbf{1},\texttt{<s>})\rangle\rangle$	$(1, \{(1,\texttt{<s>},\ 1,\texttt{<s>})\})$
$p = (2, 3,$ we must $)$, operation = replace π_1 by CONCAT(π_1, p)	
$\langle\langle(1,1,\texttt{<s>})(\mathbf{2},\mathbf{3},$ we must $)\rangle\rangle$	$(3, \{(1,\texttt{<s>},\ 3,$ must $)\})$
$p = (4, 4,$ also $)$, operation = replace π_1 by CONCAT(π_1, p)	
$\langle\langle(1,1,\texttt{<s>})(2,3,$ we must $)(\mathbf{4},\mathbf{4},$ also $)\rangle\rangle$	$(4, \{(1,\texttt{<s>},\ 4,$ also$)\})$
$p = (5, 6,$ these criticisms $)$, operation = add a new sequence $\pi_2 = \langle p\rangle$	
$\langle\langle(1,1,\texttt{<s>})(2,3,$ we must $)(4,4,$ also $)\rangle,\ \langle(\mathbf{5},\mathbf{6},$ these criticisms $)\rangle\rangle$	$(6, \{(1,\texttt{<s>},\ 4,$ also$),$ $(5,$ these$, 6,$ criticisms$)\})$
$p = (7, 7,$ seriously $)$, operation = replace π_2 by CONCAT(π_2, p)	
$\langle\langle(1,1,\texttt{<s>})(2,3,$ we must $)(4,4,$ also $)\rangle,\ \langle(5,6,$ these criticisms $)(\mathbf{7},\mathbf{7},$ seriously $)\rangle\rangle$	$(7, \{(1,\texttt{<s>},\ 4,$ also$),$ $(5,$ these$, 7,$ seriously$)\})$
$p = (8, 8,$ take $)$, operation = replace π_1, π_2 by CONCAT(π_1, p, π_2)	
$\langle\langle(1,1,\texttt{<s>})(2,3,$ we must $)(4,4,$ also $)(\mathbf{8},\mathbf{8},$ take $)(5,6,$ these criticisms $)(7,7,$ seriously $)\rangle\rangle$	$(8, \{(1,\texttt{<s>},\ 7,$ seriously$)\})$
$p = (9, 9,\texttt{</s>})$, operation = replace π_1 by CONCAT(π_1, p)	
$\langle\langle(1,1,\texttt{<s>})(2,3,$ we must $)(4,4,$ also $)(8,8,$ take $)(5,6,$ these criticisms $)(7,7,$ seriously $)(\mathbf{9},\mathbf{9},\texttt{</s>})\rangle\rangle$	$(9, \{(1,\texttt{<s>},\ 9,\texttt{</s>})\})$

Figure 1: Illustrations of how the new algorithm produces the output translation. We note each phrase that is being added and the operation it takes to generate the next segments of phrase sequence.

of target-language words $e = e_1 \dots e_m$. We use $s(p)$, $t(p)$, and $e(p)$ to refer to the three elements of a phrase p. A *derivation* is a sequence of L phrases, $p_1 \dots p_L$. The derivation gives a translation by concatenating the target-language strings $e(p_1) \dots e(p_L)$.

We will always assume that $x_1 = \texttt{<s>}$, the start-of-sentence symbol, and $x_n = \texttt{</s>}$, the end-of-sentence symbol. The only phrases covering positions 1 and n are $(1, 1, \texttt{<s>})$ and $(n, n, \texttt{</s>})$.

A derivation $p_1 \dots p_L$ is *valid* if each word in the source sentence is translated exactly once, and if for $i = 2 \dots L$ we have $|t(p_{i-1})+1-s(p_i)| \leq d$, where d is the distortion limit.

The score for any derivation is

$$f(p_1 \dots p_L) = \lambda(e(p_1) \dots e(p_L)) + \sum_{i=1}^{L} \kappa(p_i) + \sum_{i=2}^{L} \eta \times |t(p_{i-1}) + 1 - s(p_i)|$$

where the parameter η is the distortion penalty, $\lambda(e)$ is a language model score for the word sequence e, and $\kappa(p)$ is the score for phrase p under the phrase-based model. For example under a bigram language model, we have $\lambda(e_1 \dots e_m) = \sum_{i=2}^{m} \lambda(e_i|e_{i-1})$. where $\lambda(v|u)$ is the score for bigram (u, v).

The phrase-based decoding problem is to find

$$\arg\max_{p_1 \dots p_L \in \mathcal{P}} f(p_1 \dots p_L)$$

where $\mathcal{P}$ is the set of all valid derivations for the input sentence.

2.2 The Decoding Algorithm

At a high level, the decoding algorithm of Chang and Collins (2017) differs from the commonly-used approach of Koehn et al. (2003) in two important respects:

1. The decoding algorithm proceeds in strictly left-to-right order in the *source* sentence.

2. Each sub-derivation (item) in the beam consists of *multiple* sequences of phrases, instead of a single sequence.

To be more precise, each sub-derivation in the decoding algorithm consists of:

1. An integer j specifying the length of the derivation (i.e., that words $x_1 \dots x_j$ have been translated).

2. A set of segments $\{\pi_1, \pi_2, \dots, \pi_r\}$ where $r \geq 1$. Each segment π is a sequence of phrases. The segment π_1 always has $(1, 1, \texttt{<s>})$ as its first element. Each word $x_1 \dots x_j$ is translated exactly once in these segments.

As one example, the sub-derivation $(1, \{\langle(1, 1, \texttt{<s>})\rangle\})$ is always the initial sub-derivation, with only the first word x_1 being translated, and with a single segment

$\pi_1 = \langle(1, 1, \texttt{<s>})\rangle$. A more complex sub-derivation is

$$(7, \{\langle(1, 1, \texttt{<s>})(2, 3, \text{we must})(4, 4, \text{also})\rangle, \langle(5, 6, \text{these criticisms})(7, 7, \text{seriously})\rangle\}) \quad (1)$$

which translates words $x_1 \ldots x_7$, and has two segments,

$$\pi_1 = \langle(1, 1, \texttt{<s>})(2, 3, \text{we must})(4, 4, \text{also})\rangle$$
$$\pi_2 = \langle(5, 6, \text{these criticisms})(7, 7, \text{seriously})\rangle$$

We now describe how sub-derivations can be built as the source sentence is processed in left-to-right order. A derivation $(j, \{\pi_1 \ldots \pi_r\})$ can be extended as follows:

1. First select some phrase $p = (j + 1, t, e)$ where the phrase-based lexicon specifies that words $x_{j+1} \ldots x_t$ can be translated as the English sequence $e = e_1 \ldots e_m$.

2. Second, extend the derivation using one of the following operations (we use CONCAT to denote an operation that concatenates two or more phrase sequences):

   (a) Replace π_i for some $i \in 1 \ldots r$ by $\text{CONCAT}(\pi_i, p)$.

   (b) Replace π_i for some $i \in 2 \ldots r$ by $\text{CONCAT}(p, \pi_i)$.

   (c) Replace $\pi_i, \pi_{i'}$ for integers $i \neq i'$ by $\text{CONCAT}(\pi_i, p, \pi_{i'})$

   (d) Create a new segment $\pi_{r+1} = \langle p \rangle$.

Figure 1 shows the sequence of steps, and the resulting sequence of sub-derivations, in the translation of a German sentence.

A few remarks:

Remark 1. The score for each of the operations (a)-(d) described above is easily calculated using a combination of phrase, language model, and distortion scores.

Remark 2. The distortion limit can be used to rule out some of the operations (a)-(d) above, depending on the phrase p and the start/end points of each of the segments $\pi_1 \ldots \pi_r$.

Remark 3. Dynamic programming can be used with this algorithm. Under a bigram language model, the dynamic programming state for a sub-derivation $(j, \{\pi_1 \ldots \pi_r\})$ records the words and positions at the start and end of each segment $\pi_1 \ldots \pi_r$. For example under a bigram language model the sub-derivation $(7, \ldots)$ in Eq. 1 would be mapped to the dynamic-programming state $(7, \{(1, \texttt{<s>}, 4, \text{also}), (5, \text{these}, 7, \text{seriously})\})$. See Chang and Collins (2017) for more details.

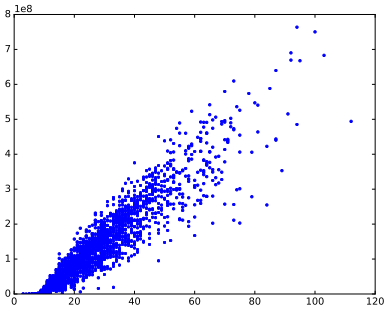

Figure 2: The total number of dynamic programming transitions and the sentence length.

Remark 4. It is simple to use beam search in conjunction with the algorithm. Different derivations of the same length j are compared in the beam. A heuristic—typically a lower-order language model—can be used to score the first $n - 1$ words in each segment $\pi_1 \cdots \pi_r$: this can be used as the "future score" for each item in the beam. This is arguably simpler than the future scores used in (Koehn et al., 2003), which have to take into account the fact that different items in the beam correspond to translations of different subsets of words in the source sentence. In our approach different derivations of the same length j have translated the same set of words $x_1 \cdots x_j$. For example in the sub-derivation $(7, \ldots)$ given above (Eq. 1), and given a trigram language model, the initial bigram *these criticisms* in π_2 is scored as $p_u(\text{these}) \times p_b(\text{criticisms}|\text{these})$ where p_u and p_b are unigram and bigram language models.

3 Experiments

The original motivation for Chang and Collins (2017) was to develop a dynamic-programming algorithm for phrase-based decoding that for a fixed distortion limit d was polynomial time in other factors: the resulting dynamic programming algorithm is $O(nd!lh^{d+1})$ time, where d is the distortion limit, l is a bound on the number of phrases starting at any position, and h is related to the maximum number of different target translations for any source position. However an open question is whether the algorithm is useful in practice when used in conjunction with beam search. This

	d	SegmentD		Segment2		Moses	
		BLEU	time	BLEU	time	BLEU	time
cs-en	8	13.67[1]	5m31s	17.42	2m49s	**17.56**	3m32s
de-en	12	25.89	9m02s	**26.69**	5m25s	**26.69**	7m37s
es-en	4	32.02	4m27s	32.01	3m58s	**32.03**	4m01s
fi-en	8	23.02	5m03s	23.66	3m09s	**23.73**	3m37s
fr-en	4	31.42	4m58s	31.43	4m23s	**31.45**	4m20s
it-en	4	28.44	4m26s	**28.44**	3m57s	28.41	3m36s
nl-en	8	24.96	7m53s	25.13	5m20s	**25.16**	5m56s
pt-en	4	**31.06**	4m28s	31.05	4m00s	31.05	3m31s
sv-en	4	31.33	3m58s	**31.35**	3m30s	31.34	3m02s
vi-en	8	20.48[2]	3m39s	**20.96**	2m08s	20.95	2m40s

Figure 3: Comparison of beam search under the new decoding algorithm and the Moses decoder. We show the BLEU score and the decoding time of three beam search based decoding methods.

# segments	# sentences	percentage
1	636	34.93%
2	1,136	62.38%
3	49	2.69%

(a) The distribution of the number of segments required for the optimal solutions. Note that the distortion limit is four.

# segments	# sentences	percentage
1	119,428	15.97%
2	541,833	72.44%
3	82,869	11.08%
4	3,747	0.50%
5	128	0.02%
6	1	0.00%

(b) The distribution of the number of segments required for reordering the parsed German sentence.

Figure 4: The number of segments required for German-to-English translation.

section describes experiments comparing beam search under the new algorithm to the method of Koehn et al. (2003). Throughout this section we refer to the algorithm of Chang and Collins (2017) as the "new" decoding algorithm.

Data. We use the Europarl parallel corpus (Version 7)[3] (Koehn, 2005) for all language pairs except for Vietnamese-English (vi-en). For Czech-English (cs-en), we use the Newstest2015 as the development set and Newstest2016 as the test set. For European languages other than Czech, we use the development and test set released for the Shared Task of WPT 2005[4]. For vi-en, we use the IWSLT'15 data.

3.1 Search Space with a Bigram Model

We first analyze the properties of the algorithm by running the exact decoding algorithm with a bigram language model and a fixed distortion limit of four, with no pruning. In Figure 2, we plot the number of transitions computed versus sentence length for translation of 2,000 German sentences to English. The figure confirms that the search space grows linearly with the number of words in the source sentence.

3.2 Beam Search under the New Algorithm

Even though the exact algorithm is linear time in the input sentence length, other factors (the dependence on d, l, and h, as described above) make the exact algorithm too costly to be useful in practice. We experiment with beam search under the new algorithm,[5] both with and without further pruning or restriction.

We experimented with a *segment constraint* on the new algorithm: more specifically, we describe experiments with a hard limit $r \leq 2$ on the number of segments $\pi_1 \ldots \pi_r$ used in any translation.

Figure 3 shows results using a trigram language model for the new algorithm with beam search (SegmentD), the new algorithm with beam search and a hard limit $r \leq 2$ on the number of segments (Segment2), and Moses. A beam size of 100 is used with all the algorithms. For each language pair, we pick the distortion limit that maximizes the BLEU score for Moses. Moses was used to train all the translation models. It can be seen that the Segment2 algorithm gives very similar performance to Moses, while SegmentD has inferior performance for languges which require a larger distortion limit.

3.3 Experiments on the Number of Segments Required for German-to-English Translation

Finally, we investigate empirically how many segments (the maximum value of r) are required for translation from German to English. In a first experiment, we use the system of Chang and Collins (2011) to give exact search for German-to-English

[1] Unable to produce translations for 36 sentences.
[2] Unable to produce translations for one sentence.
[3] http://www.statmt.org/europarl/
[4] ACL 2005 Workshop on Building and Using Parallel Texts.
[5] See Section 5.1 in Chang and Collins (2017)

translation under a trigram language model with a distortion limit $d = 4$, and then look at the maximum value for r for each optimal translation. Out of 1,821 sentences, 34.9% have a maximum value of $r = 1$, 62.4% have $r = 2$, and 2.69% have $r = 3$ (Table 4a). No optimal translations require a value of r greater than 3. It can be seen that very few translations require more than 2 segments.

In a second experiment, we take the reordering system of Collins et al. (2005) and test the maximum value for r on each sentence to capture the reordering rules. Table 4b gives the results. It can be seen that over 99% of sentences require a value of $r = 3$ or less, again suggesting that for at least this language pair, a choice of $r = 3$ or $r = 4$ is large enough to capture the majority of reorderings (assuming that the rules of Collins et al. (2005) are comprehensive).

4 Conclusion

The goal of this paper was to understand the empirical performance of a newly proposed decoding algorithm that operates from left to right on the source side. We compare our implementation of the new algorithm with the Moses decoder. The experimental results demonstrate that the new algorithm combined with beam search and segment-based pruning is competitive with the Moses decoder. Future work should consider integration of the method with more recent models, in particular neural translation models.

References

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through Lagrangian relaxation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 26–37.

Yin-Wen Chang and Michael Collins. 2017. A polynomial-time dynamic programming algorithm for phrase-based decoding with a fixed distortion limit. *Transactions of the Association for Computational Linguistics* 5:59–71.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Empirical Methods in Natural Language Processing (EMNLP)*. pages 1724–1734.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 531–540.

He He, Jordan Boyd-Graber, and Hal Daumé III. 2016. Interpretese vs. translationese: The uniqueness of human strategies in simultaneous interpretation. In *North American Association for Computational Linguistics*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1700–1709.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 48–54.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics* 30(4):417–449.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.