Cohesive Constraints in A Beam Search Phrase-based Decoder

Nguyen Bach and Stephan Vogel
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{nbach, stephan.vogel}@cs.cmu.edu

Colin Cherry
Microsoft Research
One Microsoft Way
Redmond, WA, 98052, USA
collinc@microsoft.com

Abstract

Cohesive constraints allow the phrase-based decoder to employ arbitrary, non-syntactic phrases, and encourage it to translate those phrases in an order that respects the source dependency tree structure. We present extensions of the cohesive constraints, such as exhaustive interruption count and rich interruption check. We show that the cohesion-enhanced decoder significantly outperforms the standard phrase-based decoder on English→Spanish. Improvements between 0.5 and 1.2 BLEU point are obtained on English→Iraqi system.

1 Introduction

Phrase-based machine translation is driven by a phrasal translation model, which relates phrases (contiguous segments of words) in the source to phrases in the target. This translation model can be derived from a word-aligned bitext. Translation candidates are scored according to a linear model combining several informative feature functions. Crucially, this model incorporates translation model scores and n-gram language model scores. The component features are weighted to minimize a translation error criterion on a development set (Och, 2003). Decoding the source sentence takes the form of a beam search through the translation space, with intermediate states corresponding to partial translations. The decoding process advances by extending a state with the translation of a source phrase, until each source word has been translated exactly once. Re-ordering occurs when the source phrase to be translated does not immediately follow the previously translated phrase. This is penalized with a discriminatively-trained distortion penalty. In order to calculate the current translation score, each state can be represented by a triple:

- A coverage vector HC indicates which source words have already been translated.
- A span $\bar{f}$ indicates the last source phrase translated to create this state.
- A target word sequence stores context needed by the target language model.

As cohesion concerns only movement in the source, we can completely ignore the language model context, making state effectively an $(\bar{f}, HC)$ tuple.

To enforce cohesion during the state expansion process, cohesive phrasal decoding has been proposed in (Cherry, 2008; Yamamoto et al., 2008). The cohesion-enhanced decoder enforces the following constraint: once the decoder begins translating any part of a source subtree, it must cover all the words under that subtree before it can translate anything outside of it. This notion can be applied to any projective tree structure, but we use dependency trees, which have been shown to demonstrate greater cross-lingual cohesion than other structures (Fox, 2002). We use a tree data structure to store the dependency tree. Each node in the tree contains surface word form, word position, parent position, dependency type and POS tag. We use T to stand for our dependency tree, and $T(n)$ to stand for the subtree rooted at node n. Each subtree $T(n)$ covers a span of contiguous source words; for subspan $\bar{f}$ covered by $T(n)$, we say $\bar{f} \in T(n)$.

Cohesion is checked as we extend a state $(\bar{f}_h, HC_h)$ with the translation of $\bar{f}_{h+1}$, creating a new state $(\bar{f}_{h+1}, HC_{h+1})$. Algorithm 1 presents the cohesion check described by Cherry (2008). Line 2 selects focal points, based on the last translated phrase. Line 4 climbs from each focal point to find the largest subtree that needs to be completed before the translation process can move elsewhere in the tree. Line 5 checks each such subtree for completion. Since there are a constant number of focal points (always 2) and the tree climb and completion checks are both linear in the size of the source, the entire check can be shown to take linear time.

The selection of only two focal points is motivated by a "**violation free**" assumption. If one assumes that the

Algorithm 1 Interruption Check (Coh1) (Cherry, 2008)

```
Input: Source tree $T$, previous phrase $\bar{f}_h$, current phrase $\bar{f}_{h+1}$, coverage vector $HC$
1: $Interruption \leftarrow False$
2: $F \leftarrow$ the left and right-most tokens of $\bar{f}_h$
3: for each of $f \in F$ do
4:     Climb the dependency tree from $f$ until you reach the highest node $n$ such that $\bar{f}_{h+1} \notin T(n)$.
5:     if $n$ exists and $T(n)$ is not covered in $HC_{h+1}$ then
6:         $Interruption \leftarrow True$
7:     end if
8: end for
9: Return $Interruption$
```

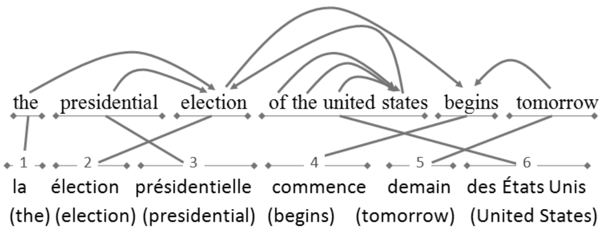

Figure 1: A candidate translation where Coh1 does not fire

translation represented by $(\bar{f}_h, HC_h)$ contains no cohesion violations, then checking only the end-points of $\bar{f}_h$ is sufficient to maintain cohesion. However, once a soft cohesion constraint has been implemented, this assumption no longer holds.

2 Extensions of Cohesive Constraints

2.1 Exhaustive Interruption Check (Coh2)

Because of the "violation free" assumption, Algorithm 1 implements the design decision to only suffer a violation penalty once, when cohesion is initially broken. However, this is not necessarily the best approach, as the decoder does not receive any further incentive to return to the partially translated subtree and complete it.

For example, Figure 1 illustrates a translation candidate of the English sentence "the presidential election of the united states begins tomorrow" into French. We consider $\bar{f}_4$ = "begins", $\bar{f}_5$ = "tomorrow". The decoder already translated "the presidential election" making the coverage vector HC_5 = "1 1 1 0 0 0 0 1 1". Algorithm 1 tells the decoder that no violation has been made by translating "tomorrow" while the decoder should be informed that there exists an outstanding violation. Algorithm 1 found the violation when the decoder previously jumped from "presidential" to "begins", and will not find another violation when it jumps from "begins" to "tomorrow".

Algorithm 2 is a modification of Algorithm 1, changing only line 2. The resulting system checks all previously covered tokens, instead of only the left and right-most tokens of $\bar{f}_{h+1}$, and therefore makes no violation-free assumption. For the example above, Algorithm 2 will inform the decoder that translating "tomorrow" also incurs a violation. Because $|F|$ is no longer constant, the time complexity of Coh2 is worse than Coh1. However, we can speed up the interruption check algorithm by hashing cohesion checks, so we only need to run Algorithm 2 once per $(\bar{f}_{h+1}, HC_{h+1})$.

Algorithm 2 Exhaustive Interruption Check (Coh2)

```
Input: Source tree $T$, previous phrase $f_h$, current phrase $f_{h+1}$, coverage vector $HC$
1: $Interruption \leftarrow False$
2: $F \leftarrow \{f | HC_h(f) = 1\}$
3: for each of $f \in F$ do
4:     Climb the dependency tree from $f$ until you reach the highest node $n$ such that $\bar{f}_{h+1} \notin T(n)$.
5:     if $n$ exists and $T(n)$ is not covered in $HC_{h+1}$ then
6:         $Interruption \leftarrow True$
7:     end if
8: end for
9: Return $Interruption$
```

Algorithm 3 Interruption Count (Coh3)

```
Input: Source tree $T$, previous phrase $\bar{f}_h$, current phrase $\bar{f}_{h+1}$, coverage vector $HC$
1: $ICount \leftarrow 0$
2: $F \leftarrow$ the left and right-most tokens of $\bar{f}_h$
3: for each of $f \in F$ do
4:     Climb the dependency tree from $f$ until you reach the highest node $n$ such that $\bar{f}_{h+1} \notin T(n)$.
5:     if $n$ exists then
6:         for each of $e \in T(n)$ and $HC_{h+1}(e) = 0$ do
7:             $ICount = ICount + 1$
8:         end for
9:     end if
10: end for
11: Return $ICount$
```

2.2 Interruption Count (Coh3) and Exhaustive Interruption Count (Coh4)

Algorithm 1 and 2 described above interpret an interruption as a binary event. As it is possible to leave several words untranslated with a single jump, some interruptions may be worse than others. To implement this observation, an interruption count is used to assign a penalty to cohesion violations, based on the number of words left uncovered in the interrupted subtree. We initialize the interruption count with zero. At any search state when the cohesion violation is detected the count is incremented by

Algorithm 4 Exhaustive Interruption Count (Coh4)

```
Input: Source tree $T$, previous phrase $f_h$, current phrase $f_{h+1}$, coverage vector $HC$
1:  $ICount \leftarrow 0$
2:  $F \leftarrow \{f | HC_h(f) = 1\}$
3:  for each of $f \in F$ do
4:      Climb the dependency tree from $f$ until you reach the highest node $n$ such that $\bar{f}_{h+1} \notin T(n)$.
5:      if $n$ exists then
6:          for each of $e \in T(n)$ and $HC_{h+1}(e) = 0$ do
7:              $ICount = ICount + 1$
8:          end for
9:      end if
10: end for
11: Return $ICount$
```

one. The modification of Algorithm 1 and 2 lead to Interruption Count (Coh3) and Exhaustive Interruption Count (Coh4) algorithms, respectively. The changes only happen in lines 1, 5 and 6. We use an additional bit vector to make sure that if a node has been reached once during an interruption check, it should not be counted again. For the example in Section 2.1, Algorithm 4 will return 4 for $ICount$ (“of”; “the”; “united”; “states”).

2.3 Rich Interruption Constraints (Coh5)

The cohesion constraints in Sections 2.1 and 2.2 do not leverage node information in the dependency tree structures. We propose the rich interruption constraints (Coh5) algorithm to combine four constraints which are Interruption, Interruption Count, Verb Count and Noun Count. The first two constraints are identical to what was described above. Verb and Noun count constraints are enforcing the following rule: a cohesion violation will be penalized more in terms of the number of verb and noun words that have not been covered. For example, we want to translate the English sentence “the presidential election of the united states begins tomorrow” to French with the dependency structure as in Figure 1. We consider $\bar{f}_h$ = “the united states”, $\bar{f}_{h+1}$ = “begins”. The coverage bit vector HC_{h+1} is “0 0 0 0 1 1 1 1 0”. Algorithm 5 will return true for $Interruption$, 4 for $ICount$ (“the”; “presidential”; “election”; “of”), 0 for $VerbCount$ and 1 for $NounCount$ (“election”).

Algorithm 5 Rich Interruption Constraints (Coh5)

```
Input: Source tree $T$, previous phrase $\bar{f}_h$, current phrase $\bar{f}_{h+1}$, coverage vector $HC$
1:  $Interruption \leftarrow False$
2:  $ICount, VerbCount, NounCount \leftarrow 0$
3:  $F \leftarrow$ the left and right-most tokens of $\bar{f}_h$
4:  for each of $f \in F$ do
5:      Climb the dependency tree from $f$ until you reach the highest node $n$ such that $\bar{f}_{h+1} \notin T(n)$.
6:      if $n$ exists then
7:          for each of $e \in T(n)$ and $HC_{h+1}(e) = 0$ do
8:              $Interruption \leftarrow True$
9:              $ICount = ICount + 1$
10:             if POS of $e$ is “VB” then
11:                 $VerbCount \leftarrow VerbCount + 1$
12:             else if POS of $e$ is “NN” then
13:                 $NounCount \leftarrow NounCount + 1$
14:             end if
15:         end for
16:     end if
17: end for
18: Return $Interruption$, $ICount$, $VerbCount$, $NounCount$
```

3 Experiments

We built baseline systems using GIZA++ (Och and Ney, 2003), Moses’ phrase extraction with grow-diag-final-end heuristic (Koehn et al., 2007), a standard phrase-based decoder (Vogel, 2003), the SRI LM toolkit (Stolcke, 2002), the suffix-array language model (Zhang and Vogel, 2005), a distance-based word reordering model with a window of 3, and the maximum number of target phrases restricted to 10. Results are reported using lowercase BLEU (Papineni et al., 2002). All model weights were trained on development sets via minimum-error rate training (MERT) (Och, 2003) with 200 unique n-best lists and optimizing toward BLEU. We used the MALT parser (Nivre et al., 2006) to obtain source English dependency trees and the Stanford parser for Arabic (Marneffe et al., 2006). In order to decide whether the translation output of one MT engine is significantly better than another one, we used the bootstrap method (Zhang et al., 2004) with 1000 samples ($p < 0.05$). We perform experiments on English→Iraqi and English→Spanish. Detailed corpus statistics are shown in Table 1. Table 2 shows results in lowercase BLEU and bold type is used to indicate highest scores. An italic text indicates the score is statistically significant better than the baseline.

	English→Iraqi		English→Spanish	
	English	Iraqi	English	Spanish
sentence pairs	654,556		1,310,127	
unique sent. pairs	510,314		1,287,016	
avg. sentence length	8.4	5.9	27.4	28.6
# words	5.5 M	3.8 M	35.8 M	37.4 M
vocabulary	34 K	109 K	117 K	173 K

Table 1: Corpus statistics

Our English-Iraqi data come from the DARPA TransTac program. We used TransTac T2T July 2007

	English→Iraqi		English→Spanish	
	july07	june08	ncd07	nct07
Baseline	31.58	23.58	33.18	32.04
+Coh1	**32.63**	24.45	33.49	*32.72*
+Coh2	32.51	**24.73**	33.52	*32.81*
+Coh3	32.43	24.19	33.37	*32.87*
+Coh4	32.32	24.66	33.47	*33.20*
+Coh5	31.98	24.42	**33.54**	***33.27***

Table 2: Scores of baseline and cohesion-enhanced systems on English→Iraqi and English→Spanish systems

(july07) as the development set and TransTac T2T June 2008 (june08) as the held-out evaluation set. Each test set has 4 reference translation. We applied the suffix-array LM up to 6-gram with Good-Turing smoothing. Our cohesion constraints produced improvements ranging between **0.5** and **1.2** BLEU point on the held-out evaluation set.

We used the Europarl and News-Commentary parallel corpora for English→Spanish as provided in the ACL-WMT 2008 shared task evaluation. The baseline system used the parallel corpus restricting sentence length to 100 words for word alignment and a 4-gram SRI LM with modified Kneyser-Ney smoothing. We used nc-devtest2007(ncd07) as the development set and nc-test2007(nct07) as the held-out evaluation set. Each test set has 1 translation reference. We obtained improvements ranging between **0.7** and **1.2** BLEU. All cohesion constraints perform statistically significant better than the baseline on the held-out evaluation set.

4 Conclusions

In this paper, we explored cohesive phrasal decoding, focusing on variants of cohesive constraints. We proposed four novel cohesive constraints namely exhaustive interruption check (Coh2), interruption count (Coh3), exhaustive interruption count (Coh4) and rich interruption constraints (Coh5). Our experimental results show that with cohesive constraints the system generates better translations in comparison with strong baselines. To ensure the robustness and effectiveness of the proposed approaches, we conducted experiments on 2 different language pairs, namely English→Iraqi and English→Spanish. These experiments also covered a wide range of training corpus sizes, ranging from 600K sentence pairs up to 1.3 million sentence pairs. All five proposed approaches give positive results. The improvements on English→Spanish are statistically significant at the 95% level. We observe a consistent pattern indicating that the improvements are stable in both language pairs.

Acknowledgments

This work is in part supported by the US DARPA TransTac programs. 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 like to thank Qin Gao and Matthias Eck for helpful conversations, Johan Hall and Joakim Nirve for helpful suggestions on training and using the English dependency model. We also thanks the anonymous reviewers for helpful comments.

References

Colin Cherry. 2008. Cohesive phrase-based decoding for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 72–80, Columbus, Ohio, June. Association for Computational Linguistics.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP'02*, pages 304–311, Philadelphia, PA, July 6-7.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL'07*, pages 177–180, Prague, Czech Republic, June.

Marie-Catherine Marneffe, Bill MacCartney, and Christopher Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC'06*, Genoa, Italy.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC'06*, Genoa, Italy.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 1:29, pages 19–51.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL'03*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL'02*, pages 311–318, Philadelphia, PA, July.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904, Denver.

Stephan Vogel. 2003. SMT decoder dissected: Word reordering. In *Proceedings of NLP-KE'03*, pages 561–566, Bejing, China, Oct.

Hirofumi Yamamoto, Hideo Okuma, and Eiichiro Sumita. 2008. Imposing constraints from the source tree on ITG constraints for SMT. In *Proceedings of the ACL-08: HLT, SSST-2*, pages 1–9, Columbus, Ohio, June. Association for Computational Linguistics.

Ying Zhang and Stephan Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of EAMT'05*, Budapest, Hungary, May. The European Association for Machine Translation.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *Proceedings of LREC'04*, pages 2051–2054.