Elastic-substitution decoding for Hierarchical SMT: efficiency, richer search and double labels

Gideon Maillette de Buy Wenniger[1] gemdbw AT gmail.com
Khalil Sima'an[2] k.simaan AT uva.nl
Andy Way[1] andy.way AT adaptcentre.ie

[1]ADAPT Centre, School of Computing, Dublin City University, Dublin, Ireland
[2]Institute for Logic Language and Computation (ILLC), Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands

Abstract

Elastic-substitution decoding (ESD), first introduced by Chiang (2010), can be important for obtaining good results when applying labels to enrich hierarchical statistical machine translation (SMT). However, an efficient implementation is essential for scalable application. We describe how to achieve this, contributing essential details that were missing in the original exposition. We compare ESD to strict matching and show its superiority for both reordering and syntactic labels. To overcome the sub-optimal performance due to the late evaluation of features marking label substitution types, we increase the diversity of the rules explored during cube pruning initialization with respect to labels their labels. This approach gives significant improvements over basic ESD and performs favorably compared to extending the search by increasing the cube pruning pop-limit. Finally, we look at combining multiple labels. The combination of reordering labels and target-side boundary-tags yields a significant improvement in terms of the word-order sensitive metrics Kendall reordering score and METEOR. This confirms our intuition that the combination of reordering labels and syntactic labels can yield improvements over either label by itself, despite increased sparsity.

1 Introduction

Elastic-substitution decoding (ESD) – also known as *soft label matching* or *soft-constraint decoding* – is an effective method to gain maximal benefit from the use of labels to enrich hierarchical phrase-based statistical machine translation (SMT), and was first introduced by Chiang (2010). This method removes many of the disadvantages of working with labeled grammars when labels are strictly enforced. We discuss the requirements and details of an efficient implementation in the first part of this paper, to benefit other researchers that want to apply ESD. In the second part of the paper we further strengthen the empirical evidence for the success of ESD. This is done by comparing strict and soft-labeled (ESD) systems for Chinese–English translation, using four different types of labels. Next, we describe how the results of ESD can be further improved for small label sets by diversifying the search, exploring all alternatively labeled versions of each rule source-side type during cube pruning initialization instead of only the single best one. This is compared against the more crude approach of just increasing the search space by increasing the cube pruning *pop-limit*. Finally, we explore the effect of combining multiple labels, either the two types of reordering labels or a reordering

label with a syntactic label. All source code for both ESD and labeled grammar extraction is made publicly available with this publication.[1]

2 Background and Related Work

Hierarchical phrase-based SMT (or hierarchical SMT for short) (Chiang, 2005) is the hierarchical generalization of phrase-based SMT (Koehn et al., 2003). It generalizes phrase-pairs into synchronous context-free grammar (SCFG) rules by adding variables to them. This yields a weighted SCFG (Aho and Ullman, 1969). The particular form of SCFGs used in this paper is called HIERO (Chiang, 2005), and allows only up to two nonterminals (variables) in the right-hand-side of rules. This gives the following four HIERO rule types:

$$X \rightarrow \langle \alpha,\ \delta \rangle \tag{1}$$

$$X \rightarrow \langle \alpha\ X_{\boxed{1}} \gamma,\ \delta\ X_{\boxed{1}} \eta \rangle \tag{2}$$

$$X \rightarrow \langle \alpha\ X_{\boxed{1}} \beta\ X_{\boxed{2}} \gamma\ ,\ \delta\ X_{\boxed{1}}\ \zeta\ X_{\boxed{2}} \eta\ \rangle \tag{3}$$

$$X \rightarrow \langle \alpha\ X_{\boxed{1}} \beta\ X_{\boxed{2}} \gamma\ ,\ \delta\ X_{\boxed{2}}\ \zeta\ X_{\boxed{1}} \eta\ \rangle \tag{4}$$

Here $\alpha, \beta, \gamma, \delta, \zeta, \eta$ are terminal sequences that can be empty, except for β, since HIERO prohibits rules with nonterminals that are adjacent on the source side. HIERO additionally requires all rules to have at least one pair of aligned words. These extra constraints are intended to reduce the amount of spurious ambiguity. Equation (1) corresponds to a normal phrase pair, (2) to a rule with one gap and (3) and (4) to the monotone and inverting rules, respectively.

In addition, HIERO has a special glue rule: (g1) $GOAL \rightarrow \langle GOAL_{\boxed{1}}\ X_{\boxed{2}},\ GOAL_{\boxed{1}}\ X_{\boxed{2}} \rangle$ as well as two special start/end rules: (g2) $GOAL \rightarrow \langle GOAL_{\boxed{1}}\ <\!/s\!>\ ,\ GOAL_{\boxed{1}}\ <\!/s\!> \rangle$ and (g3) $GOAL \rightarrow \langle <\!s\!>, <\!s\!> \rangle$, with $<\!s\!>$ and $<\!/s\!>$ being the dedicated start/end symbols.

HIERO makes very strong independence assumptions, since it uses only one label "X" apart from the glue symbol *GOAL*, allowing any HIERO rule to substitute to any other rule. A lot of work has been done on relaxing these assumptions by labeling HIERO with labels derived from syntax (Zollmann and Venugopal, 2006; Almaghout et al., 2011), dependency information (Li et al., 2012), word classes such as POS-tags (Zollmann and Vogel, 2011), reordering information (Maillette de Buy Wenniger and Sima'an, 2014) and other types of information.

However, labeling with strict matching of labels splits the rules of HIERO into many alternatively labeled variants, increasing spurious ambiguity. Venugopal et al. (2009) introduced preference grammars as a way to avoid this increase and to relax the assumptions of decoding with strict matching. Every rule is equipped with label distributions instead of single labels, for both the left- and right-hand-side rule nonterminals. Using a dynamic programming approach, these label distributions are then multiplied during decoding, to approximate the probability over the full set of alternatively labeled derivations. Unlike preference grammars, ESD does not approximate selection of the most likely unlabeled derivation. However, in contrast it can learn to treat different substitutions such as NP → NPP differently from others such as NP → VP which the formalism of preference grammars cannot, as it lacks a learning component. This is a clear advantage for heuristically created labels such as syntax-augmented machine translation (SAMT) and others used in this paper.

[1] Source code URLs: *ESD*: https://github.com/gwenniger/joshua/commits/gideon/cubePruningFixForFuzzyMatching
Grammar extraction: https://bitbucket.org/gwenniger/labeled-translation https://bitbucket.org/teamwildtreechase/hatparsing

The work by Chiang (2010) on ESD, the foundation of the work in this paper, is discussed next.

3 Elastic-substitution decoding

ESD was introduced by Chiang (2010), who describes it as follows: "*... we allow any rule to substitute into any site, but let the model learn which substitutions are better than others.*"

With respect to decoding it is remarked that: "*The decoding algorithm then operates as in hierarchical phrase-based translation. The decoder has to store in each hypothesis the source and target root labels of the partial derivation, but these labels are used for calculating feature vectors only and not for checking well-formedness of derivations.*"

In summary ESD entails:

(A) Adapting the decoder to support soft-matching of labels, which means finding all matching rules while ignoring the labels.

(B) Adding *label-substitution features* that mark different types of substitutions: (i) matching and mismatching substitutions, and (ii) substitutions of particular types of labels to particular gaps to enable learning what type of substitutions are preferable.

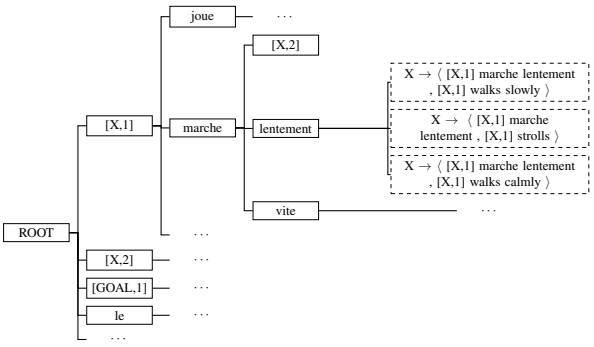
To enable computing the label-substitution features (B), the labels must be left present in the hypergraph (the packed hypotheses) computed by the decoder.

3.1 ESD: a naive implementation

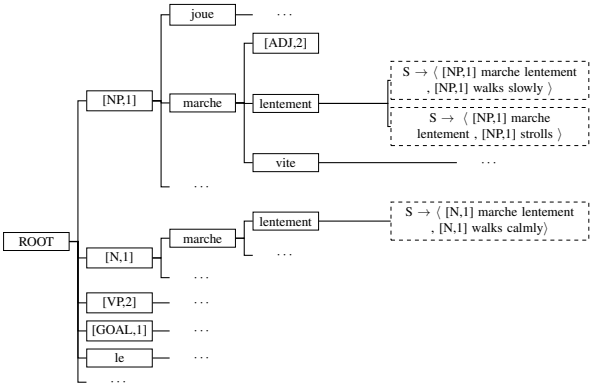
With strict matching, the inner loop of the decoder finds all matching rules r_{match} for an input word span $s_{<i,j>} = w_i \dots w_j$. For the rule right-hand-side $rhs = RHS(r_{match})$, given the ordered words $w_k \in rhs$ and nonterminals $nt_l \in rhs$, w_k must match the corresponding word in $s_{<i,j>}$ and nt_l must match the label of a corresponding chart span that was previously covered by the decoder (so-called "rule gap"); both must be matched in accordance with the input order. Adding ESD to this process, a naive implementation explicitly matches all best alternatively labeled rule variants for an (unlabeled) source-rule type, to all alternatively labeled gaps. This naive implementation is, however, computationally expensive. For rules with up to two gaps the number of source rule variants can increase quadratically with the size of the label set N, and analogously the same holds for the two substituted-to gaps of these rules. Hence this naive approach gives an increase in computational complexity of $O(N^4)$.

3.2 How is ESD implemented efficiently?

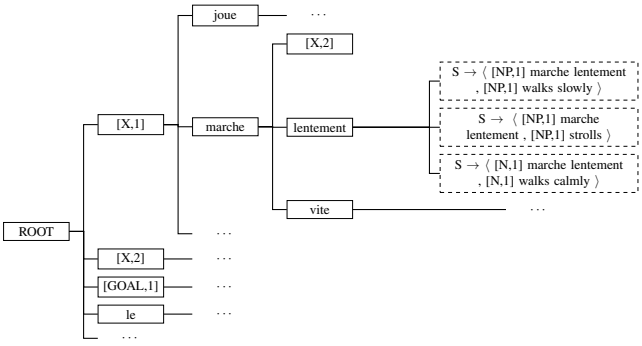
During normal (strict) decoding, matching rules are found through lookup in a dedicated rule indexing data-structure called a *trie* (De La Briandais, 1959). An efficient ESD implementation requires adaptation of this trie, rather than explicitly generating all types of label matches. Figure 1 shows rule tries used by the decoder to find matching rules during decoding, for three cases: (a) HIERO, (b) labeled system with strict matching, and (c) labeled system with ESD. Note that for (a) there are no labels except the default label "X" and the glue rule label "GOAL". In (b), labels are present both in the internal nodes and also in the leaf nodes containing the complete rules. Note that the rules "$S \rightarrow \langle NP_{\square}$ *marche lentement,* $NP_{\square}$ *walks slowly*$\rangle$" and "$S \rightarrow \langle N_{\square}$ *marche lentement,* $N_{\square}$ *walks calmly*$\rangle$" which are identical on the source-side except for their right-hand side nonterminal label (NP versus N), have distinct paths in the trie. For ESD, labels are not used as constraints and therefore need to be removed. This allows the decoder to quickly find all matching rules for a sequence of nonterminals and lexical items, without unnecessarily splitting the trie into many paths for different labelings. However, when during cube-pruning complete rules are added to the chart, the labels should still be obtainable from



(a) Rule trie for HIERO.



(b) Rule trie for a labeled system with strict label matching.



(c) Rule trie for a labeled system with soft label matching (ESD).

Figure 1: Rule tries for three different system types.

them because they are required for computing label-substitution features. This is achieved by keeping the labels outside the trie nodes but retaining them inside the rules that are stored in lists at each leaf node in the trie. This is exactly what is done in the trie for ESD (Figure 1 (c)). As an illustration, note how in (c) the rules "$S \rightarrow \langle NP_{\boxed{1}}$ *marche lentement,* $NP_{\boxed{1}}$ *walks slowly*$\rangle$" and "$S \rightarrow \langle N_{\boxed{1}}$ *marche lentement,* $N_{\boxed{1}}$ *walks slowly*$\rangle$" share the same unlabeled path in the trie as in (a), but their labels are still retained in the complete rules stored at the leaf node.

During decoding, ESD extends hypotheses with all rules matching the source input, while ignoring labels. This is done by substituting the actual labels from the hypergraph with surrogate "X" labels and using those labels to retrieve matching rules from the rule trie. There is, however, an important exception to this that requires special treatment, namely the nonterminal label occurring in glue rules. Glue rules of the form

$$GOAL \rightarrow \langle GOAL_{\boxed{1}}\, X_{\boxed{2}},\ GOAL_{\boxed{1}}\, X_{\boxed{2}} \rangle$$

contain two types of labels. The "X" symbol in the rule is the symbol that will be substituted to HIERO (non-glue) rules. The $GOAL$ label, occurring on the left-hand side of the rule and as the first nonterminal on the right-hand side, is also known as the start symbol. It serves to start the gluing extension and allows for the glue rule to be used repeatedly. This $GOAL$ label needs to be strictly matched, to prevent the left-hand side of glue rules from softly matching other nonterminals and hence substituting for HIERO rules. The strict matching of the $GOAL$ label is achieved in the grammar by retaining it as a label in the trie used by ESD (see the "GOAL" labeled internal node in Figure 1 (c), the third child of ROOT), and requiring the $GOAL$ symbols observed in the hypergraph to be strictly matched against the symbols in the trie. Furthermore, labels inside HIERO rules should not be allowed to match the $GOAL$ label but only the surrogate label X that represents the rest of the labels, when retrieving matching rules from the trie. This implementation ensures correct and efficient rule matching given either $GOAL$ labels (strict-matching) or other labels (ESD).

One other important detail enabling efficient ESD decoding is that the used labeled ESD grammars are identical in size to HIERO. Let the HIERO-*rule-signature* of a labeled rule be that rule with the labels removed. Given a rule labeling scheme, grammars used with ESD are formed by labeling every HIERO rule with a single *canonical* labeling: the most frequent labeled version across extracted rules that share the HIERO-*rule-signature* of that rule. These grammars also use the same feature set as HIERO, only adding *label-substitution features*. In contrast, because strict matching systems combine all differently labeled extracted rule versions, they use grammars that are much bigger than HIERO grammars.

4 Experiments

We evaluate our models on Chinese–English, since it facilitates the best comparison with experiments in earlier work. All data is lowercased as a last pre-processing step. The training data for our experiments is formed by combining the full sentence-aligned *MultiUN* (Eisele and Chen, 2010; Tiedemann, 2012)[2] parallel corpus with the full sentence-aligned *Hong Kong Parallel Text* parallel corpus from the Linguistic Data Consortium.[3] We used a maximum sentence length of 40 for filtering the training data. The combined dataset has 7,340,000 sentence pairs. For the development and test sets we use the *Multiple-Translation Chinese* datasets from LDC, parts 1–4,[4] which contain sentences from the News domain. We combined parts 2 and 3 to form the development set (1,813 sentence pairs) and parts 1 and 4 to form the

[2]Freely available from http://opus.lingfil.uu.se/

[3]The LDC catalog number of this dataset is LDC2004T08.

[4]LDC catalog numbers: LDC2002T01, DC2003T17, LDC2004T07 and LDC2004T07.

test set (1,912 sentence pairs). For both development and testing we use 4 references. For these experiments both the baseline and our method use a 4-gram language model with Kneser-Ney smoothing (Kneser and Ney, 1995) trained on 5,427,696 sentences of *domain-specific*[5] news data taken from the "Xinhua" subcorpus of LDC's English Gigaword corpus.[6]

4.1 Training and decoding details

Our experiments use Joshua (Ganitkevitch et al., 2012) with Viterbi best derivation. Baseline experiments use normal decoding, whereas ESD experiments relax the label-matching constraints while adding label-substitution features to facilitate learning of label-substitution preferences.

For training we use standard HIERO grammar extraction constraints (Chiang, 2007) (phrase pairs with source spans up to 10 words; abstract rules are forbidden). During decoding a maximum span of 10 words on the source side is maintained. In our experiments, for HIERO we use a standard feature set that is comparable to that of Chiang (2005). We follow Chiang (2010) in using, except for the label-substitution features, exactly the same features for ESD as for HIERO. This includes the usage of phrase-weights taken from the HIERO (label-stripped) rules as opposed to the labeled rules. For the labeled systems with strict matching (-Str), we follow Zollmann (2011) in using phrase weights for the labeled versions of the rules, but also adding smoothed versions of these features, including the HIERO (unlabeled) phrase weights. We train our systems using (batch k-best) MIRA (Cherry and Foster, 2012) as borrowed by Joshua from the Moses codebase, allowing up to 30 tuning iterations. Following standard practice, we tune on BLEU (Papineni et al., 2002), and after tuning we use the configuration with the highest scores on the development set with actual (corpus-level) BLEU evaluation. We report lowercase BLEU, METEOR (Denkowski and Lavie, 2011), BEER (Stanojević and Sima'an, 2014) and TER (Snover et al., 2006) scores for the test set. We also report average translation length as a percentage of the reference length for all systems.

To counter unreliable conclusions due to optimizer variance, we repeated all experiments three times (tuning plus testing), and compute the scores as averages over these runs; using Multeval Clark et al. (2011) version 0.5.1.[7] We also use MultEval's implementation of statistical significance testing between systems, which is based on multiple optimizer runs and approximate randomization. Differences that are statistically significant with respect to a HIERO baseline and correspond to improvement/worsening are marked with $\triangle H/\triangledown H$ at the $p \leq .05$ level and $\blacktriangle H/\blacktriangledown H$ at the $p \leq .01$ level. For average translation length, where either higher or lower may be better, we use $\square H/\blacksquare H$ to mark significant *change* with respect to the baseline at the $p \leq .05$ / $p \leq .01$ level.

We also report the Kendall reordering score (KRS), which is the reordering-only variant of the LR-score (Birch et al., 2010) (without the optional interpolation with BLEU) and which is a sentence-level score. For the computation of statistical significance of this metric we use our own implementation of the *sign test* (Dixon and Mood, 1946), described also by Koehn (2010).

Finally we report the average CPU time per translated sentence in the test set. These times are obtained using special Java system methods, and aggregated over all decoder threads and the main thread. These time statistics are robust to variations in the number of decoder threads and the amount of other jobs running on the server, factors that can easily confound statistics based on regular wall-clock time.

[5]The different domain of the training data (mainly parliament) and development/test data (news) requires usage of a domain-specific language model to obtain optimal results.

[6]The LDC catalog number of this dataset is LDC2003T05.

[7]https://github.com/jhclark/multeval

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length	CPU time
HIERO	31.63	30.56	13.15	**59.28**	58.03	97.15	3.34
HIERO-0^{th}-Str	31.90 ▲H	30.79 ▲H	13.45	60.11 ▼H	59.68 ▲H	98.65 ■H	2.87
HIERO -0^{th}	32.03▲H	30.70▲H▼S	13.42▲H	59.58▼H▲S	58.87▲H▼S	97.87■H■S	8.99
HIERO-1^{st}-Str	31.77	30.62	13.20	60.13 ▼H	59.89 ▲H	98.47 ■H	4.63
HIERO -1^{st}	32.35▲H▲S	30.98▲H▲S	13.75▲H▲S	60.26▼H	60.01▲H	99.11■H▲S	8.45
SAMT-Str	31.87 △H	30.61	13.38	59.97 ▼H	59.94 ▲H	98.46 ■H	25.59
SAMT	32.40▲H▲S	31.20▲H▲S	14.01▲H▲S	60.19▼H▽S	60.38▲H△S	99.37■H■H	8.09
BoundaryTag-Str	32.26▲H	30.94▲H	13.91▲H	60.20▼H	58.78▲H	**98.98**▲H	29.29
BoundaryTag	**32.77**▲H▲S	**31.27**▲H▲S	**14.17**▲H▲S	60.15▼H	**60.83**▲H▲S	99.72■H■S	8.60

Table 1: Results for labeled systems with strict or soft label matching. Statistical significance is given against the HIERO baseline (H) and pair-wise for every soft-matching system against its strict-matching variant (-Str). Statistical significance for the latter comparison is marked with (S). For every experiment we use boldface to accentuate the highest score across systems for all metrics, with for TER, an error metric, the lowest score instead. For length we boldface the value that is closest to 100, in absolute terms.

4.2 Is soft label matching always superior to strict matching?

In Table 1 we compare four labeled systems for decoding with strict matching and decoding with soft label matching. This extends the earlier comparison by Maillette de Buy Wenniger and Sima'an (2014, 2016), attempting to give a more general answer to the question as to whether soft label matching is always superior to strict matching. The first two systems are reordering labeled systems (Maillette de Buy Wenniger and Sima'an, 2013, 2014, 2016), and the last two systems are syntactically labeled systems, namely SAMT (Zollmann and Venugopal, 2006) and a target-side boundary-tag labeled system (Zollmann, 2011; Zollmann and Vogel, 2011).

Label Types: Our reordering labels are heuristic labels, created using hierarchical reordering information induced from word alignments. These labels come in two forms:1) 0^{th}-order reordering labels (HIERO-0^{th}) describe for each nonterminal the reordering that happens at its child nonterminals, 2) 1^{st}-order reordering labels (HIERO-1^{st}) describe the reordering of the nonterminal itself relative to an embedding parent nonterminal. SAMT is a heuristic syntactic labeling scheme, similar in spirit to combinatory categorial grammar (CCG) (Steedman, 1987, 2000). SAMT uses constituency-parse information and finds the simplest syntactic label describing a (target) span. Similar to SAMT, target-side boundary-tags are heuristic syntactic labels formed by combining the POS-tags of (target) words at the boundaries of phrase pairs. Since limited space allows only a short description of the used labels and systems, we refer the reader to the original papers for more details.

For three of the four label types tested, the soft-labeled system gives significantly better scores for BLEU, METEOR and BEER. Only the HIERO-0^{th} label type does not show significantly better results for those metrics. However, later in this section we discuss how these results too are improved by extending the search. Although it is not statistically significant, HIERO-0^{th} still shows improved BLEU for the soft-labeled version over the strict matching system. For SAMT and the target-side boundary-tag labeled system, apart from the other improvements, there are also significant improvements of KRS. The results show that soft matching is typically, although not always significantly, better than strict matching.

4.3 Challenges of soft label matching

In the previous section we saw that decoding with soft label matching typically outperforms both unlabeled systems and systems that use labels with strict matching. Nevertheless, efficient soft label matching faces two challenges: (i) increased search space, and (ii) label matching blindness.

The addition of labels increases the search space dramatically, even with soft matching.

During decoding, a rule is applied to extend an existing hypothesis. Since in practice decoding proceeds bottom up, this means any right-hand-side nonterminal label(s) of the rule are matched with the labels of the corresponding substituted to nonterminal gaps in the chart. With N labels, there are there are potentially N alternative versions per language model state in the chart entries of each gap. For a rule with 2 gaps, this means a particular rule substitution only covers one of up to N^2 possible options arising from the splitting of the language model states by labeling. Because soft matching keeps only one labeled version per HIERO-*rule-signature*, on the rule side the number of options does not increase compared to HIERO.[8]

The second challenge, which we call *label matching blindness*, is that the type of applied label substitutions (and whether or not these are matching) is only evaluated late in the search process. During initialization, cube pruning explores the combination of the best rule with the best leaf nodes for the matched-to rule gaps in the chart. However, the quality of the rule and the leaf nodes is only computed based on local (stateless) features, such as lexical probabilities and phrase weights. Features such as the language model cost cannot be computed because they cross rule boundaries. They may still be approximated given the available information, but this is generally inaccurate. In the case of *label-substitution features*, no meaningful stateless computation is possible. These features are therefore simply ignored until the *rule-nonterminal to labeled-gap* substitutions have already been decided during cube-pruning initialization.

4.4 Enriching the search

The challenges resulting from soft label matching mentioned in the last section motivate enrichment of the search during soft label matching decoding. A crude approach is to just extend the search space by increasing the value of the decoder parameter *pop-limit*, which controls the number of hypotheses that are added to the stack by cube pruning during decoding. This may improve the quality of the produced translations at the price of a higher computational cost. However, this approach is computationally expensive and inefficient, since it does not directly target an exploration of rule-to-gap substitutions with diverse labels. The initial label substitution diversity is determined by the diversity of label pairs within the set of *rule-RHS-nonterminal to chart-gap* substitutions explored during cube-pruning *initialization*. For small label sets it is feasible to enrich the set of those initially explored substitutions, thereby drastically increasing this diversity. Three ways to implement this are:

a) Exploring all alternatively labeled versions of a HIERO *source* rule type.[9]

b) Exploring all alternatively labeled gap substitutions, given the single best labeled version of a HIERO *source* rule type.

c) Combining (a) and (b), i.e. exploring all labeled rule versions and for each of those all alternatively labeled gap substitutions.

Why should this help? Assume that we explore all alternatively labeled rule versions for a HIERO rule type (a), while keeping the gap labels fixed to those of the best language model state. Then, a matching substitution will be explored if yielded by substituting the nonterminal labels for any of these alternatively labeled rule versions to those fixed gap labels. Similarly for

[8]However, with larger label sets, the canonical labeled rule form represents a larger number of rules, and will consequently be a more approximate representation of those.

[9]Note that whereas there is only one canonical labeled version per HIERO-*rule-signature*, there are potentially many labeled versions of the *source* rule type, in which the target side is ignored. Furthermore, the LHS label of the rule is ignored when collecting the best alternatively labeled versions given a HIERO rule *source* side, since it has no effect on label matching in bottom-up decoding.

System Name	Pop-limit	Explore all rule labelings	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length	CPU time
HIERO	1000	——–	31.63	30.56	13.15	59.28	58.03	97.15	3.34
HIERO 0^{th} (B_0)	1000	NO	32.03▲H	30.70▲H	13.42▲H	59.58▼H	58.87▲H	97.87■H	8.99
HIERO 0^{th}	2000	NO	32.24▲H▲B_0	30.66▲H	13.41	**59.01**▲H▲B_0	58.59△H	97.42■H■B_0	16.13
HIERO 0^{th}	4000	NO	32.30▲H▲B_0	30.85▲H▲B_0	13.71▲H▲B_0	59.56▼H	59.23▲H△B_0	98.19■H■B_0	28.27
HIERO 0^{th}	1000	YES	32.18▲H	30.77▲H△B_0	13.55▲H△B_0	59.23 ▲B_0	58.74▲H	97.69■H■B_0	9.32
HIERO 0^{th}-Sh	1000	YES	32.25▲H▲B_0	30.80▲H▲B_0	13.60▲H▲B_0	59.44▽H	59.12▲H	97.99■H	10.14
HIERO 0^{th}-Sh	2000	YES	32.47▲H▲B_0	30.87▲H▲B_0	13.69▲H▲B_0	59.51▼H	59.27▲H△B_0	98.27■H■B_0	16.59
HIERO 0^{th}-Sh	4000	YES	32.26▲H▲B_0	30.77▲H△B_0	13.55▲H△B_0	59.12△H▲B_0	58.72▲H	97.55■H■B_0	37.75
HIERO 1^{st} (B_1)	1000	NO	32.35▲H	30.98▲H	13.75▲H	60.26▼H	**60.01**▲H	99.11■H	8.45
HIERO 1^{st}	2000	NO	32.40▲H	31.00▲H	13.74▲H	60.36▼H	59.93▲H	99.15■H	15.40
HIERO 1^{st}	4000	NO	32.49▲H	30.95▲H	13.75▲H	60.29▼H	60.00▲H	99.15■H	28.84
HIERO 1^{st}	1000	YES	32.50▲H△B_1	31.03▲H△B_1	13.80▲H	60.14▼H	59.77▲H	99.05■H	9.55
HIERO 1^{st}-Sh	1000	YES	32.66▲H▲B_1	**31.05**▲H▲B_1	**13.88**▲H	60.01▼H▲B_1	59.99▲H	99.14■H	10.16
HIERO 1^{st}-Sh	2000	YES	32.62▲H▲B_1	30.90▲H▼B_1	13.70▲H	59.72▼H▲B_1	59.41▲H▼B_1	98.60■H■B_1	21.79
HIERO 1^{st}-Sh	4000	YES	**32.70**▲H▲B_1	30.99▲H	13.83▲H	60.22▼H	59.91▲H	**99.20**■H	40.81

Table 2: The effects on the results of: 1) the pop-limit, 2) exploring all alternative rule labelings, with or without shuffling. Statistical significance is given against the HIERO baseline (H) and for each label against the basic search version: HIERO 0^{th} without additional search (B_0) or HIERO 1^{st} without additional search (B_1).

System Name	Pop-limit	Explore all rule labelings	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length
HIERO	1000	——–	31.70	30.72	13.40	61.21	58.28	99.49
HIERO 0^{th} (B_0)	1000	NO	31.85▲H	30.82▲H	13.78▲H	61.46▼H	59.08▲H	**100.00**■H
HIERO 0^{th}	2000	NO	32.08▲H	30.76	13.77▲H	**60.86**▲H	58.92▲H	99.61
HIERO 0^{th}	4000	NO	32.03▲H	30.97▲H	13.94▲H	61.47▼H	59.42▲H	100.36■H
HIERO 0^{th}	1000	YES	32.05▲H	30.86▲H	13.72△H	61.09	59.05▲H	99.93■H
HIERO 0^{th}-Sh	1000	YES	32.01▲H	30.92▲H	13.82▲H	61.33	59.53▲H	100.19■H
HIERO 0^{th}-Sh	2000	YES	32.12▲H	30.97▲H	13.93▲H	61.42▽H	59.53▲H	100.23■H
HIERO 0^{th}-Sh	4000	YES	32.07▲H	30.90▲H	13.83▲H	60.99△H	58.95▲H	99.85■H
HIERO 1^{st} (B_1)	1000	NO	32.13▲H	31.22▲H	14.21▲H	62.31▼H	**60.61**▲H	101.29■H
HIERO 1^{st}	2000	NO	32.43▲H	31.24▲H	14.24▲H	62.20▼H	60.43▲H	101.33■H
HIERO 1^{st}	4000	NO	32.46▲H	31.26▲H	14.30▲H	62.02▼H	**60.61**▲H	101.35■H
HIERO 1^{st}	1000	YES	32.33▲H	31.22▲H	14.32▲H	62.12▼H	60.32▲H	101.15■H
HIERO 1^{st}-Sh	1000	YES	32.38▲H	31.27▲H	**14.37**▲H	62.05▼H	60.29▲H	101.19■H
HIERO 1^{st}-Sh	2000	YES	32.48▲H	31.09▲H	14.10▲H	61.65▼H	59.85▲H	100.73■H
HIERO 1^{st}-Sh	4000	YES	**32.55**▲H	**31.30**▲H	**14.37**▲H	62.20▼H	60.55▲H	101.49■H

Table 3: Analysis: results on the **development** set. We demonstrate the effect on the results of: 1) the pop-limit, 2) exploring all alternative rule labelings, with or without shuffling. Statistical significance is given against the HIERO baseline (H).

Evaluation order without shuffling:

1. S -> [NP,1] marche lentement, [NP,1] walks slowly (A)
2. S -> [NP,1] marche lentement, [NP,1] strolls (B)
3. S -> [N,1] marche lentement, [N,1] walks calmly (C)
4. **S -> Elle marche [ADV,1], She walks [ADV,1] (D)**

Final order after scoring:

1. S -> [NP,1] marche lentement, [NP,1] walks slowly |||
   Elle marche lentement / She walks slowly ||| -4.0 (A)
2. S -> [NP,1] marche lentement, [NP,1] strolls |||
   Elle marche lentement / She strolls ||| -5.0 (B)
3. S -> [N,1] marche lentement, [N,1] walks calmly |||
   Elle marche lentement / She walks calmly ||| -5.0 (C)
4. **S -> Elle marche [ADV,1], She walks [ADV,1]**
   Elle marche lentement / She walks leisurely ||| -5.0 (D)

(a) Evaluation order and final order without shuffling.

Evaluation order with shuffling (i.e. random):

1. **S -> Elle marche [ADV,1], She walks [ADV,1]** (D)
2. S -> [NP,1] marche lentement, [NP,1] walks slowly (A)
3. S -> [N,1] marche lentement, [N,1] walks calmly (C)
4. S -> [NP,1] marche lentement, [NP,1] strolls (B)

Final order after scoring:

1. S -> [NP,1] marche lentement, [NP,1] walks slowly |||
   Elle marche lentement / She walks slowly ||| -4.0 (A)
2. **S -> Elle marche [ADV,1], She walks [ADV,1]**
   Elle marche lentement / She walks leisurely ||| -5.0 (D)
3. S -> [N,1] marche lentement, [N,1] walks calmly |||
   Elle marche lentement / She walks calmly ||| -5.0 (C)
4. S -> [NP,1] marche lentement, [NP,1] strolls |||
   Elle marche lentement / She strolls ||| -5.0 (B)

(b) Evaluation order and final order with shuffling.

Figure 2: Example of the effect of shuffling on the decoding, translating the source phrase "Elle marche lentement".

(b), given a single fixed labeled rule version, a matching substitution is explored if achievable in combination with the available gap labelings.

With respect to computational complexity, (a) and (b) potentially increase the number of explored combinations by a factor of N^2 with N being the size of the label set, whereas (c) even increases it by N^4. These increases are with respect to the number of applicable HIERO rule types, since the increased exploration only concerns the cube pruning initialization and not the whole cube pruning process. Consequently, if N is small, the empirical increase in computational cost is limited. With larger N however, the increase in computational cost of the initialization (quadratic for (a) and (b) and N^4 for (c)), starts to dominate the total cost, so that none of these approaches scale to large[10] label sets, with (c) scaling up worst.

Ie what follows, we only explore (a), which is very similar to (b). We will refer to this setting as diverse rule labeling exploration (DRLE) in the rest of the paper. We do not explore (c) here, because of its very restricted scalability.

4.5 Shuffling

In cube-pruning initialization, applicable rules are substituted to particular gaps, and every complete rule substitution leads to a total score that includes the label-substitution features and language-model cost amongst other things. These complete rule substitutions are then added as initial options to the cube-pruning queue.

In our initial implementation of DRLE (see Section 4.4), we consecutively evaluated all alternative labeled versions for a specific HIERO source-rule type before moving on to the next type. Independent of this evaluation order, rules are placed on the cube-pruning queue in the order of their evaluation scores. Nevertheless, we discovered that it helps if we shuffle the order of the rules before we evaluate them. This might seem odd, since this shuffling (*-Sh*) can only affect the order of rules yielding the same score. However, that is exactly how we think shuffling helps; without shuffling, all labeled versions of the same rule source-side with the same score are lumped together in the cube pruning queue. Shuffling mixes them with other rules that tie for the same score. This increases diversity when neither the labels nor the translation for a rule source-side are discriminative, as is common for certain rules at the start of tuning when

[10]The approaches were still feasible for at least $N = 25$ labels, the highest number we have tested. We acknowledge however, that once N increases significantly, problems will be encountered due to computational complexity.

label-substitution feature weights are initialized to zero.

To better understand shuffling and how it effects the order of hypotheses in the cube-pruning queue, it is helpful to look at an example. In Figure 2 we show the effect of shuffling on the translation of the French source phase "Elle marche lentmentent". Here, we assume that the first source word ("Elle") and last source word ("lentement") have already been translated before. Hence, a full translation can be formed by combining these previous translations with HIERO rules that additionally translate the first two words ("Elle marche") or last two words ("marche lentement") respectively, substituting the earlier translated last/first word as a gap. In Figure 2a we show the evaluation order and final order of hypotheses without shuffling. As mentioned before, without shuffling all rules that share the same source-side (ignoring labels) – in this case rules A,B and C – are evaluated consecutively. Then in the next step the scored rules are sorted by their score, which in this example does not further change the order. In Figure 2b in contrast, the rules are shuffled, randomly permuting their order before evaluation. In this case the random order of rule evaluation is D,A,C,B. Then after scoring, rule A again comes to the top, because it has the highest score. The relative order of D,C and B however remains changed because rules B, C and D tie for the same score, so their relative order before scoring determines their relative final order. Notice in particular how rule D (in boldface), which has a different source side from rules A, B and C, now comes directly after rule A in the final order.

Note that shuffling only randomizes the relative order in which rules tying for the same score are added to the cube-pruning queue, eliminating implementation-specific bias for the order of such rules. This avoids different labeled versions of the same rule with the same score all clinging together in the queue as an undesirable side-effect of the specifics of the DRLE implementation. Because shuffling randomizes the order of rules with the same score in the final queue, it also removes the opportunity for the tuner to lazily exploit partially deterministic order in development set hypotheses which is of no use for translation of the test set. Possibly, this by itself also has a positive effect in making tuning more robust and reducing the chance of overfitting. We leave it for future work to further investigate this. Crucially, shuffling does not specifically add additional search errors.

4.6 Effects of search extension strategies

Table 2 shows the effects of the different strategies described in the previous sections to expand the search space. The table first repeats the results for the HIERO baseline and then lists results for the HIERO 1^{st} and HIERO 0^{th} reordering labels. For each we then use either the standard setting for rule exploration during cube-pruning initialization, or DRLE. We do this in combination with shuffling, and also vary the pop-limit. This provides insight into the effect of these factors when applied independently or combined.

In addition to the HIERO baseline, our second baseline (B_0/B_1) for each of the two separate reordering labels is a basic ESD system without changes to the default pop-limit (1000) and without DRLE. For each reordering label we then test the significance of improvements against both HIERO and B_0/B_1. We see that both increasing the pop-limit and DRLE improves results for both label types. However, when only the pop-limit is increased without DRLE, HIERO 0^{th} improves significantly over the basic (B_0) system, while HIERO 1^{st} gives no significant improvements over the basic (B_1) system. In addition, for HIERO 1^{st}, comparing the effect of just using DRLE to just changing the pop-limit, the former outperforms the latter for all metrics except KRS over all tested values of the pop-limit. Adding shuffling makes this trend is even sharper. For both label types, shuffling has a positive effect on the results. Note too that DRLE has the highest impact with larger label sets; probably the resulting larger search space increases the chance of missing certain desirable label substitutions in the normal cube pruning initialization.

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length	CPU time
HIERO	31.63	30.56	13.15	**59.28**	58.03	97.15	3.34
HIERO 0^{th}	32.50▲H	30.88▲H	13.68▲H	59.66▼H	59.30▲H	98.40■H	29.50

Table 4: Analysis experiment: tuning with pop-limit 2000 and test-set decoding with pop-limit 4000.

Summarizing the results over both label types, we can conclude that DRLE is typically better than just crudely increasing the pop-limit. Additionally, for small label sets it comes at a lower computational cost.

4.7 Analysis: negative interaction DRLE with higher pop-limit explained.

Both DRLE and an increased pop limit by themselves have a positive effect on the translation quality, but a surprising result is the sometimes relatively negative effect of using DRLE and also maximally increasing the pop-limit. With HIERO 0^{th} the results improve when increasing the pop-limit to 2000, but then drop when further increasing it to 4000. For HIERO 1^{st}, with DRLE only BLEU benefits slightly from a pop-limit higher than 1000, whereas performance decreases for the other metrics. Such negative interactions between DRLE and the increased pop-limit could be caused by overfitting.[11] To see if overfitting indeed occurs, we looked at the evaluation scores for the development set, see Table 3. It can be seen that for HIERO 0^{th} with DRLE, the BLEU scores decrease when the pop-limit increases from 2000 and 4000, but in particular the decrease in the development set BLEU score is less than the decrease in the test set BLEU score, see Table 2. Furthermore, when looking at HIERO 1^{st} with DRLE on the development set, it can be seen that the BLEU score monotonically increases for increasing pop-limit size. However, other metrics show a dip for a pop-limit of 2000, which was also seen for the test set for all metrics except TER. To summarize, for certain increases in the pop-limit in combination with DRLE, we made two observations that indicate overfitting:

- Loss of performance on the test set, for most metrics including BLEU, the tuning metric.
- Mostly retained or even increased performance for BLEU on the development set, combined with performance loss for most other metrics.

Could it still be that a higher pop-limit is by itself harmful, independent of its role in the assumed overfitting? We hypothesize that it is only harmful in as far as it facilitates overfitting in combination with DRLE during the tuning process. To test this hypothesis we ran another analysis experiment, whereby we use the final feature weights obtained from tuning with DRLE with a pop-limit of 2000 and only increase the pop-limit to 4000 during the decoding of the test set. The results are shown in Table 4. As can be seen, in this setting the results are highly similar to the results obtained with DRLE and a pop-limit of 2000 used for both tuning and testing. This confirms our hypothesis that a higher pop-limit (more search) is not generally harmful, but can be harmful in the tuning stage because it facilitates more overfitting.

4.8 Combining Labels

In this section we look at the effect of combining multiple labels. The first successful combination we explore is 0^{th}-order and 1^{st}-order reordering labels. Since both labels individually give good results, and encode somewhat different information about word order, their combination could work even better. The other two combinations we test are 1^{th}-order reordering labels combined with SAMT or target-side boundary-tags. These combinations are

[11] DRLE groups the translations of the source side by their labels and uses the best translation for each distinct labeling, as opposed to only a single best translation. This may cause also more suboptimal source rule-side translations to be added to the initial cube-pruning queue; and with a higher pop-limit there is a higher chance of those being retained and causing problems such as overfitting.

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length	CPU time
HIERO	31.63	30.56	13.15	**59.28**	58.03	97.15	3.34
HIERO 0^{th} + HIERO 1^{st}	32.30▲H▲H_0	30.95▲H▲H_0	13.75▲H▲H_0	60.16▼H▼H_0	60.13▲H▲H_0	99.07▲H■H_0	7.18
SAMT+ HIERO 1^{st}	32.57▲H△H_1	31.07▲H▼S▲H_1	13.84▲H▽S	59.94▼H▲H_1	60.18▲H	99.13■H■S	11.63
Bnd.Tags+ HIERO 1^{st}	**32.65**▲H▲H_1	**31.36**▲H▲B▲H_1	**14.16**▲H▲H_1	60.21▼H	**61.46**▲H▲B▲H_1	**99.86**■H■H_1	10.32

Table 5: Double-labeled systems with soft matching. Result are for exploring only the best rule labeling and label substitution during cube pruning initialization. Statistical significance is given against the HIERO baseline (H) and for every double-labeled system against the single-labeled systems from which the double label is composed: HIERO 0^{th} (H_0), HIERO 1^{st} (H_1), SAMT (S) and target-side boundary-tags (B).

intuitively promising since reordering labels and syntactic labels are expected to give at least partially different information that may be expected to be complementary.

When combined labels are directly applied to form label-substitution features, this yields a quadratic increase in the number of these features, causing extreme sparsity and hence overfitting problems. We thus take another approach; during feature generation, we split every combined label into its two constituent parts and compute individual label-substitution features for each. Table 5 shows the results of the label-combination experiments. Most of the double-labeled systems come to the level of the best of the two constituent labels, but do not improve beyond it. However, the system that combines target-side boundary-tags and 1^{th}-order reordering labels significantly improves over both these labels individually for both METEOR and KRS. These metrics are particularly concerned with assessing the quality of the word order, which receives less or no attention in the other metrics. Since reordering labels are particularly expected to improve word order, it is positive that they help to further improve it for the best-performing single label in our experiments.

5 Conclusion

In this work, we examined key aspects of effective and efficient ESD. We first gave a detailed description of how this method can be efficiently implemented, and then examined three empirical questions. First, based on experiments for four different label types, we demonstrated that ESD is empirically at least equal but typically superior to strict matching. Next, we demonstrated that ESD can benefit from richer search. Our experiments show that it is more effective to specifically target the search effort towards the exploration of more diverse label substitutions instead of crudely increasing search in general by using a higher pop-limit. Finally, we explored the effect of double labels, and showed that while these are not successful in general, the specific combination of target-side boundary-tags and reordering labels does significantly improve word order as measured by METEOR and KRS, without significantly changing the other metrics.

Acknowledgements

This research is supported by the ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106). This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Marie Skłodowska-Curie grant agreement No 713567. The investigations were supported by The Netherlands Organization for Scientific Research (NWO) under grant nr. 612.066.929 and VICI grant nr. 277-89-002 and Stichting voor de Technische Wetenschappen (STW) grant nr. 12271. We would like to thank the anonymous reviewers for their helpful comments.

References

Aho, A. V. and Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.

Almaghout, H., Jiang, J., and Way, A. (2011). CCG contextual labels in hierarchical phrase-based smt. In *Proceedings of the 15th Annual Conference of the European Association for Machine Translation (EAMT-2011)*, pages 281–288, Leuven, Belgium.

Birch, A., Osborne, M., and Blunsom, P. (2010). Metrics for MT evaluation: Evaluating reordering. *Machine Translation*, 24(1):15–26.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montreal, Canada.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Ann Arbor, Michigan.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chiang, D. (2010). Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1443–1452, Uppsala, Sweden.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, pages 176–181, Portland, Oregon.

De La Briandais, R. (1959). File searching using variable length keys. In *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference*, IRE-AIEE-ACM '59 (Western), pages 295–298, San Francisco, California.

Denkowski, M. and Lavie, A. (2011). Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland.

Dixon, W. J. and Mood, A. M. (1946). The statistical sign test. *Journal of the American Statistical Association*, 41(236):557–566.

Eisele, A. and Chen, Y. (2010). Multiun: A multilingual corpus from united nation documents. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 2868–2872, La Valletta, Malta.

Ganitkevitch, J., Cao, Y., Weese, J., Post, M., and Callison-Burch, C. (2012). Joshua 4.0: Packing, pro, and paraphrases. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 283–291, Montréal, Canada.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 181–184, Detroit, Michigan, USA.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P., Och, F. J., and Marcu, D. (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*, pages 48–54, Edmonton, Canada.

Li, J., Tu, Z., Zhou, G., and van Genabith, J. (2012). Using syntactic head information in hierarchical phrase-based translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 232–242, Montréal, Canada.

Maillette de Buy Wenniger, G. and Sima'an, K. (2013). Hierarchical alignment decomposition labels for hiero grammar rules. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 19–28, Atlanta, Georgia, USA.

Maillette de Buy Wenniger, G. and Sima'an, K. (2014). Bilingual markov reordering labels for hierarchical SMT. In *Proceedings of the Eight Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 11–21, Doha, Quatar.

Maillette de Buy Wenniger, G. and Sima'an, K. (2016). Labeling hiero grammars without linguistic resources. *Machine Translation*, pages 1–41.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA.

Stanojević, M. and Sima'an, K. (2014). BEER: BEtter evaluation as ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA.

Steedman, M. (1987). Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403–439.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 2868–2872, Istanbul, Turkey.

Venugopal, A., Zollmann, A., Smith, N. A., and Vogel, S. (2009). Preference grammars: softening syntactic constraints to improve statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 236–244, Boulder, Colorado.

Zollmann, A. (2011). *Learning Multiple-Nonterminal Synchronous Grammars for Statistical Machine Translation*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Zollmann, A. and Venugopal, A. (2006). Syntax augmented machine translation via chart parsing. In *NAACL 2006 - Workshop on statistical machine translation*, pages 138–141, New York City, New York, USA.

Zollmann, A. and Vogel, S. (2011). A word-class approach to labeling pscfg rules for machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1–11, Portland, Oregon, USA.