The Prague Bulletin of Mathematical Linguistics
NUMBER 91 JANUARY 2009 67–78

Grammar based statistical MT on Hadoop
An end-to-end toolkit for large scale PSCFG based MT

Ashish Venugopal, Andreas Zollmann

Abstract

This paper describes the open-source Syntax Augmented Machine Translation (SAMT) [1]on Hadoop toolkit—an end-to-end grammar based machine statistical machine translation framework running on the Hadoop implementation of the MapReduce programming model. We present the underlying methodology of the SAMT approach with detailed instructions that describe how to use the toolkit to build grammar based systems for large scale translation tasks.

1. Introduction

1.1. PSCFG approaches to Machine Translation

Syntax Augmented Machine Translation (SAMT) (Zollmann and Venugopal, 2006) defines a specific parameterization of the probabilistic synchronous context-free grammar (PSCFG) approach to machine translation. PSCFG approaches take advantage of nonterminal symbols, as in monolingual parsing, to generalize beyond purely lexical translation. Consider the example rule below:

$$\text{@VP} \rightarrow \text{ne @VB}_1 \text{ pas \# do not @VB}_1 : w$$

representing the discontiguous translation of the French words "ne" and "pas" to "do not", in the context of the labeled nonterminal symbol "@VB" (representing the syntactic constituent type of Verb). These rules seem considerably more complex than weighted word-to-word rules (Brown et al., 1993), or phrase-to-phrase rules (Koehn, Och, and Marcu, 2003, Och and Ney, 2004) but can be viewed as natural extensions to these well established approaches. An introduction to PSCFG approaches to machine translation can be found in (Chiang and Knight, 2006).

[1]Released under the GNU Lesser General Public License, version 2

Please cite this article as: Ashish Venugopal, Andreas Zollmann, Grammar based statistical MT on Hadoop: An end-to-end toolkit for large scale PSCFG based MT. The Prague Bulletin of Mathematical Linguistics No. 91, 2009, 67–78.

(Chiang, 2005) describes a procedure to learn PSCFG rules from word-aligned parallel corpora, using the phrase-pairs from (Koehn, Och, and Marcu, 2003) as a lexical basis for the grammar. SAMT (Zollmann and Venugopal, 2006) extends the procedure from (Chiang, 2005) to assign labels to nonterminal symbols based on target language phrase structure parse trees.

In this paper, we describe an end-to-end statistical machine translation framework—SAMT on Hadoop—to learn and estimate parameters for PSCFG grammars from word-aligned parallel corpora (*training*), and perform translation (*decoding*) with these grammars under a log-linear translation model (Och and Ney, 2004). While our framework specifically implements (Chiang, 2005) and (Zollmann and Venugopal, 2006), the training and decoding algorithms in our toolkit can be easily replaced to experiment with alternative PSCFG parameterizations like (Galley et al., 2006, Wu, 1997). The algorithms in this toolkit are implemented upon Hadoop (Cutting and Baldeschwieler, 2007), an open-source implementation of the MapReduce (Dean and Ghemawat, 2004) framework, which supports distribution computation on large scale data using clusters of commodity hardware. We report empirical results that demonstrate the use of the SAMT toolkit on large scale translation tasks.

1.2. The SAMT toolkit

Our toolkit, when used in concert with other open-source components and publicly available corpora, contains all of the necessary components to build and evaluate grammar based statistical machine translations systems. The primary components of the toolkit are listed below:

- A top level push-button script that provides experimental work-flow management and submits jobs to the underlying Hadoop framework.
- Components to build and estimate parameters for the grammars described in (Chiang, 2005) and (Zollmann and Venugopal, 2006).
- Tools to filter large translation grammars and n-gram language models to build small sentence specific models that can be easily loaded into memory during decoding.
- A bottom-up dynamic chart parsing decoder based on (Chappelier and Rajman, 1998) which supports grammars with more than 2 nonterminals symbols per rule. The decoder outputs n-best lists with optional annotations that facilitate discriminative training.
- An implementation of Minimum Error Rate (MER) training (Och, 2003), extended to perform feature selection.

The SAMT toolkit requires the following inputs that are easily generated by existing open-source tools.

- Word aligned parallel corpora. For small resource tasks, word-alignments can be generated using the GIZA++ toolkit (Och and Ney, 2003), while large-resource tasks can be aligned using (Dyer et al., 2008), a parallelized GIZA++ implementation on MapReduce.
- (Zollmann and Venugopal, 2006) requires target language parse trees for each sentence in the training data. SAMT on Hadoop interfaces to the parser from (Charniak, 2000) to parse the target side of the parallel corpora on Hadoop.
- N-Gram language models built via the SRILM toolkit (Stolcke, 2002) are used as features

during decoding.

1.3. SAMT on Hadoop

The SAMT toolkit is built upon Hadoop (Cutting and Baldeschwieler, 2007), an open-source implementation of the MapReduce model to distribute the estimation of PSCFG grammars and to perform decoding. Training and decoding are broken up into a series of MapReduce tasks, called *phases*, which are performed sequentially, transforming input data into a PSCFG grammar, and using the grammar to translate development and test sentences. Phase outputs are stored on the Hadoop Distributed File System (HDFS), a highly fault tolerant file system that is accessible by all cluster machines. Most SAMT phases are run sequentially, using output from previous phases as input[2]. Detailed instructions for downloading and building the SAMT toolkit are available at the toolkit's website[3], along with examples that that can be used to re-generate published results from (Zollmann, Venugopal, and Vogel, 2008). In the remainder of this paper, we describe the SAMT methodology and important user parameters in our toolkit that impact translation quality and runtime. For a more formal description of the individual MapReduce phases in the SAMT pipeline, see (Zollmann, Venugopal, and Vogel, 2008).

2. Syntax Augmented Machine Translation

2.1. Phrase and SAMT Rule Extraction

In this section, we describe Syntax Augmented Machine Translation (SAMT) (Zollmann and Venugopal, 2006), a specific instantiation of the PSCFG formalism that is implemented in the SAMT on Hadoop toolkit. SAMT extends the purely hierarchical grammar proposed in (Chiang, 2005) to use nonterminal labels learned from target language parse trees. The inputs to the SAMT rule extraction procedure are tuples, $\langle f, e, \mathtt{Phrases}(a, f, e), \pi\rangle$, where f is a source sentence, e is a target sentence, a is a word-to-word alignment associating words in f with words in e, $\mathtt{Phrases}(a, e, f)$, are the set of phrase pairs (source and target phrases) consistent with the alignment a (Koehn, Och, and Marcu, 2003, Och and Ney, 2004), and π is a phrase structure parse tree of e. SAMT rule extraction associates each phrase pair from $\mathtt{Phrases}(a, e, f)$ with a left-hand-side label, and then applies the rule extraction procedure from (Chiang, 2005) to generate rules with *labeled* nonterminal symbols.

Consider the example alignment graph (a word alignment and target language parse tree as defined in (Galley et al., 2006)) for the example French-to-English sentence in Figure 1. The phrase extraction method from (Koehn, Och, and Marcu, 2003), extracts all phrase pairs where no word inside the phrase pair is aligned to a word outside the phrase pair. Figure 2 gives the initial rules extracted for our example sentence pair.

[2]While these scripts assume the Hadoop-on-Demand machine requisitioning model, the toolkit can be easily modified to submit jobs to a single global machine pool

[3]www.cs.cmu.edu/~zollmann/samt

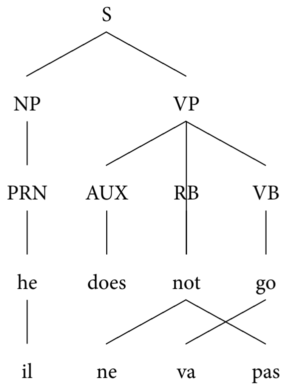

Figure 1. Alignment graph (word alignment and target parse tree) for a French-English sentence pair.

PRP:NP	$\rightarrow$	il # he
VB	$\rightarrow$	va # go
RB+VB	$\rightarrow$	ne va pas # not go
VP	$\rightarrow$	ne va pas # does not go
S	$\rightarrow$	il ne va pas # he does not go

Figure 2. Labeled initial rules.

S	$\rightarrow$	PRP:NP_1 ne va pas # PRP:NP_1 does not go
S	$\rightarrow$	il ne VB_1 pas # he does not VB_1
S	$\rightarrow$	il VP_1 # he VP_1
S	$\rightarrow$	il RB+VB_1 # he does RB+VB_1
S	$\rightarrow$	PRP:NP_1 VP_2 # PRP:NP_1 VP_2
S	$\rightarrow$	PRP:NP_1 RB+VB_2 # PRP:NP_1 does RB+VB_2
VP	$\rightarrow$	ne VB_1 pas # does not VB_1
RB+VB	$\rightarrow$	ne VB_1 pas # not VB_1
VP	$\rightarrow$	RB+VB_1 # does RB+VB_1

Figure 3. Generalized rules.

Phrase Extraction is the first phase of the SAMT toolkit, annotating each sentence-pair of the training corpus with a set of phrase pairs extracted from that sentence pair. We use a single toolkit binary: **MapExtractPhrases**, run as Hadoop Map step (there is no Reduce step in this phase). This binary takes a single numberical argument which determines the maximum length of the initial phrase extracted from word-aligned data. This limit has an impact on the size and nature of the final grammar. Typically, phrase limits are significantly smaller than the length of the parallel sentence, preventing very long distance reordering effects from being captured in the grammar.

The next phase, Rule Extraction includes rule indetification (Map step, binary **MapExtractRules**) on a per-sentence basis, and merging and counting of identical rules (Reduce step, binary **MergeRules**). SAMT assigns a left-hand-side (lhs) label to every phrase pair extracted from the current sentence-pair, based on the corresponding target language parse tree π, forming *initial rules*. These labels are assigned based on the constituent spanning the target side word sequence in π. When the target side of the phrase-pair is spanned by a single constituent in π, the constituent label is assigned as the lhs for the phrase pair. If the target side of the phrase is not spanned by a single constituent in π, we use the labels of subsuming, subsumed, and neighboring constituents in π to assign an extended label of the form $C_1 + C_2$, C_1/C_2, or $C_2\backslash C_1$ (similar in motivation to the labels in (Steedman, 1999)), indicating that the phrase pair's target side spans two adjacent syntactic categories (e.g., *she went: NP+VB*), a partial syntactic category C_1 missing a C_2 at the right (e.g., *the great: NP/NN*), or a partial C_1 missing a C_2 at the left (e.g., *great wall: DT\NP*), respectively. The label assignment is attempted in the order just described, i.e., assembling labels based on '+' concatenation of two subsumed constituents is preferred, as smaller constituents tend to be more accurately labeled. If no label is assignable by either of these three methods, and the parameter '*-allow_double_plus 1*' is set, we try triple-concatenation to create a label of the form $C_1 + C_2 + C_3$. If this approach do not yield a label or if '*-allow_double_plus 0*', a default label '_FAIL' is assigned. An ambiguity arises when unary rules $N_1 \rightarrow \ldots \rightarrow N_m$ in the target parse tree are encountered, such as the NP→PRN subtree in Figure 1. Depending on the parameter '*-unary_category_handling*', we use the bottom-most label (parameter value '*bottom*'), the top-most ('*top*'), or a combined label $N_m : \ldots : N_1$ ('*all*', this is the default).

An alternative method of assigning labels to phrase pairs can be activated by specifying the parameter '*-use_only_pos*'. In this variant, labeling is performed merely based on the part-of-speech (POS) tags of the first word *POS1* and last word *POS2* of the target phrase, resulting in the label '*POS1-POS2*'. In general, the SAMT approach can take advantage of any labeling techniques that assigns labels to arbitrary initial phrase pairs. Alternative techniques could include using source language constituent labels, or automatically induced labels. Based on these initial rules, we perform the rule *generalization* procedure from (Chiang, 2005). Figure 3 shows the resulting generalized rules. For each labeled rule in the grammar, we can also generate a corresponding generically labeled rule as in (Chiang, 2005). We introduce an additional feature in the log-linear translation model that allows the decoder to prefer labeled or unlabeled derivations. To suppress the creation of generic rules, pass the parameter '*-generate_generic_variant 0*'.

The number of rules generated by this procedure is exponential in the number of initial phrases pairs, producing a grammar that is impractical for efficient translation. The following parameters are used to restrict the number of rules extracted per sentence:

- *-max_abstraction_count* (default: 2): maximum number of abstractions (nonterminal pairs) per rule.
- *-max_source_symbol_count* (default: 6): maximum number of symbols (terminals and nonterminals) on the source side of the rule.

This restricted rule set can be pruned further with the following parameters for **MergeRules**:

- *-allow_consec_nts* (default: 1): if set to 0, discards rules that have consecutive nonterminals on the source side.
- *-allow_src_abstract* (default: 1): if 0, discards rules that do not have any source terminal symbols for example: $S \rightarrow NP_1 VP_2 \# NP_2 VP_1$. Setting this parameter to 0, drastically reduces decoding time.
- *-nonlexminfreq, -lexminfreq* (defaults: 0): minimum occurrence frequency thresholds for non-lexical and lexical rules respectively. Increasing these thresholds reduces the size of the grammar, but often at the cost of translation quality (Zollmann et al., 2008).
- *-min_freq_given_src_arg* (default: 0): minimum relative frequency of a rule given its labeled source.

The labeling and extraction procedures defined above identify rules from the input word-aligned parallel corpora and associated parse trees. The occurrence counts from this extraction process are used in estimating relative frequency features for each rule. The estimation of these features is described in the next section.

2.2. PSCFG Features

Given a source sentence f and a PSCFG grammar, the translation task can be expressed analogously to monolingual parsing with a CFG. We find the most likely derivation D of the input source sentence and read off the English translation, identified by composing α from each rule used in the derivation. This search for the most likely derivation can be defined as:

$$\hat{e} = \text{tgt}\left(\underset{D \in Derive(G):src(D)=f}{\arg\max} p(D)\right) \tag{1}$$

where $\text{tgt}(D)$ refers to the sequence of target terminal symbols generated by the derivation D, $\text{src}(D)$ refers to the source terminal symbols of D and $Derive(G)$ is the set of sentence spanning derivations of grammar G. The distribution p over derivations is defined by a log-linear model. The probability of a derivation D is defined in terms of the rules r that are used in D:

$$p(D) = \frac{p_{\text{LM}}(\text{tgt}(D))^{\theta_{\text{LM}}} \prod_{r \in D} \prod_{i=1}^{m} \lambda_i(r)^{\theta_i}}{Z(\theta_{\text{LM}}, \theta_1, \ldots, \theta_m)} \tag{2}$$

where $\lambda_i(r)$ refers to features defined on each rule, p_{LM} is an n-gram language model (LM) probability distribution over target word sequences, and Z is a normalization constant that does not need to be computed during search under the arg max search criterion in Equation 1. The feature weights $\theta_{\text{LM}}, \theta_1, \ldots, \theta_m$ are trained in concert with the language model weight via MER training. The features $\lambda_i(r)$ are statistics estimated from rule occurrence counts.

The output of the Rule Extraction phase is a grammar with a small subset of features in λ that has been learned automatically from the input data. The features used in the our toolkit include those in (Chiang, 2005, Zollmann and Venugopal, 2006), and are computed in the Rule Extraction and Filtering phase (described below). The resulting grammar is large, and for most translations tasks, cannot be loaded directly into memory for decoding. To avoid this problem, the SAMT toolkit filters the grammar against a specific test corpus, generating a sentence specific grammar for each *sentence* in the corpus. This filtering is performed for each corpora that we need for translation, typically *development*, *test*, and *unseen test* corpora are used to train and evaluate machine translation systems.

2.3. Rule and LM Filtering

The Rule Filtering phase (binaries **MapSubsampleRules**, **filterrules_bin**) take as input: the grammar from the Rule Extraction phase, a corpus to filter the grammar against, and additional model files (such as translation lexica) to generate additional rule features in λ. In the

Map step, the grammar is filtered on a per-sentence basis by matching the source words of each rule to the source words in the sentence we want to translate. In the Reduce step, additional features (documented in **filterrules.pl**, which is used to generate the MapReduce binary **filterrules_bin**). The Reduce step of Rule Filtering provides several options to further restrict the grammar and to augment the additional features. These options can be specified via the top-level parameter: *filter_params*. The Rule Filtering Reduce step also adds the following system rules to each sentence specific grammar.

- Beginning-of-sentence rule: $S \rightarrow \langle s \rangle \# \langle s \rangle$
- Glue rules (Chiang, 2005) for each NT N in the grammar, for example: $S \rightarrow S_1 N_2 \# S_1 N_2$
- End-of-sentence rule: $S \rightarrow S_1 \langle \backslash s \rangle \# S_1 \langle \backslash s \rangle$
- 'Unknown'-rules (e.g. NNP→ _UNKNOWN # _UNKNOWN) generating a limited set of labels for the word '_UNKNOWN', which the decoder substitutes for unknown source words

The Glue rules (Chiang, 2005) play an important role in grammar based approaches to MT. These rules serve to simply concatenate translations of consecutive spans during decoding, similar to monotone decoding in a phrase based system (Koehn, Och, and Marcu, 2003). These Glue operations allow the system to produce translations that violate the syntactic contraints encoded in the labels of the grammar—at a cost determined via the MER trained weight $\theta_{\mathtt{glue}}$.

Building sentence specific grammars allows us to estimate the parameters and features of the grammar on large parallel corpora, while still being able to load all relevant rules to translate particular sentences in a test corpus. We follow this same approach to filter large n-gram language models in a LM Filtering phase. While the Rule Filtering phase filters rules based on the source side of the rule, the n-gram LM must be filtered according to the possible set of target words that can generated by applying the sentence specific grammar. For each sentence specific grammar, a possible target vocabulary is generated, which is used by the Rule Filtering binary (**LMFilter**) to produce sentence specific language models.

3. PSCFG Decoding

The runtime complexity of our decoder with an integrated n-gram LM feature is:

$$\mathcal{O}\left(|f|^3 \left[|\mathcal{N}||\mathcal{T}_T|^{2(n-1)}\right]^K\right) \tag{3}$$

where K is the maximum number of NT symbols per rule, $|f|$ is the source sentence length, $\mathcal{N}$ is the set of nonterminal labels in the grammar, $\mathcal{T}_T$ is the set of target language terminals the in grammar, and n is the order of the n-gram LM. Our decoder implements the Cube Pruning algorithm from (Chiang, 2007), and outputs n-best lists for use in MER. The **FastTranslateChart** performs translation as a Map task. The grammar restriction parameters described in Section 2.1 have a large significant impact on decoding runtime (particulary *allow_src_abstract, allow_consec_nts, max_abstraction_count*), but this search still requires additional pruning to produce translations in reasonable time-frames—especially when translating longer sentences. The most important decoder parameters are described below:

- *wts*: corresponds to the weights θ in the translation model in Equation 2. In practice, these weights are iteratively trained via MER.
- *HistoryLength*: (default 2) The number of words considered as LM history length during decoding. When set to less than $n-1$, when using an n-gram LM, decoding time is reduced at the expense of search errors, which can reduce translation quality.
- *SRIHistoryLength*: This value indicates the full history length of the n-gram language model. When using a reduced *HistoryLength*, this value is used to recover from search errors in a LM-driven n-best extraction step similar to (Huang and Chiang, 2007).
- *PruningMap*: (default: 0-100-5-@_S-200-5): Format: lhs-b-β. Pruning parameters for Cube Pruning (Chiang, 2007). For each nonterminal label lhs in the grammar for a source span during decoding, this parameter restricts the number of chart items to b items, and items that are have cost of at most β greater than the best item. lhs $= 0$ sets pruning parameters for all lhs symbols that have not been explicly specified.
- *ComboPruningBeamSize* : (default 10000) Sets the maximum number of items generated in each cell via Cube Pruning. Reducing this value reduces decoding time when *PruningMap* limits have not caused pruning.
- *MaxHypsPerCell*: (default 1000000000) Limits the total number of items (partial translation hypotheses) created for each span during decoding—across items that have different lhs labels (not counting X and S items, which always pass thru this pruning filter). This value is typically set when using grammars with a large number of lhs labels to reduce translation runtime, but does introduce additional search error.
- *MaxCostDifferencePerCell*: (default inf) Max. allowed cost that an item can deviate from the best item in its chart cell (inf: any cost allowed). Items with lhs X or S always pass thru this filter. This and the previous parameter are the only paramters that apply pruning across items with different nonterminal labels.
- *MaxCombinationCount*: (default 10) Limits the application of automatically learned PSCFG rules to source spans less than or equal to *MaxCombinationCount*. Spans of greater length are composed monotonically with Glue rules. Decoding time is linear in sentence length once this limit is in effect.

3.1. Minimum Error Rate Training

The parameters θ are trained via MER training to maximize translation quality according to a user specified automatic translation metric, like BLEU (Papineni et al., 2002) or NIST (Doddington, 2002). MER training is implemented in the SAMT toolkit as a MapReduce phase using n-best lists from the decoding phase. Our MER implementation performs feature selection, preferring solutions where $\theta_i = 0$, and can be easily extended to perform random restarts as well.

Track	Words (English)	LM 1-N grams (N)	Dev.	Test1	Test2
IWSLT	632K	431,292 (5)	IWSLT06	IWSLT07	N/A
67M	67M	102,924,025 (4)	MT05	MT06	MT08
230M	230M	273,233,010 (5)	MT05	MT06	MT08

Table 1. Training data configurations used to evaluated SAMT on Hadoop. The number of words in the target text and the number of 1-N grams represented in the complete model are the defining statistics that characterize the scale of each task. For each LM we also indicate the order of the n-gram model.

System	Dev. BLEU	Test1 BLEU	Test2 BLEU	Grammar Train. (h:m)	Test1 (m)
IWSLT Hier	27.0	37.0	N/A	0:12	4
IWSLT Syntax	30.9	37.2	N/A	0:26	12
67M Hier	35.19	32.98	25.88	1:10	17
67M Syntax	35.69	33.12	26.48	2:26	65
230M Hier	36.39	33.74	26.28	4:13	23
230M Syntax	37.11	34.04	26.74	7:21	53

Table 2. Translation quality as measured by IBM-BLEU% (i.e., brevity penalty based on closest reference length) on each resource track for appropriate evaluation data sets. Systems 67M and 230M are evaluated in lower-case, while IWSLT is evaluated in mixed case. Training and decoding times given are based on a cluster of 100 (2 per machine) 1.9GHz Intel Xeon processors.

4. Empirical Results

We demonstrate the SAMT on Hadoop toolkit on three Chinese-to-English translation tasks, representing a wide range of resource conditions. Each task is described in Table 1. The IWSLT task is a limited resource, limited domain task, while 67M and 230M (named for their respective corpora sizes), are corpora used for the annual NIST MT evaluation. For each task we list the number of words in the target side of the corpus and the number of 1-n grams in the n-gram LM (estimated from parallel and monolingual data).

For each resource condition, we build SAMT systems using a purely hierarchical grammar (Hier) (Chiang, 2005) and a syntax augmented grammar (Syntax) from (Zollmann and Venugopal, 2006). All experiments use a 2-gram *HistoryLength* length the first pass of decoding, and the full LM history during the second pass n-best list search. These grammars are built with '*-allow_consec_nts 0 -allow_src_abstract 0*', and the NIST MT task rules are additionally restricted by '*-nonlexminfreq 2 -min_freq_given_src_arg* α' where $\alpha = 0.005$ (Hier) and $\alpha = 0.01$ (Syntax). The Syntax based systems also use '*-MaxHypsPerCell 1000*' to limit the run time impact of the large number of lhs labels in these grammars.

In Table 2, we report BLEU scores on development and test data as well as run times to train the respective PSCFG grammars and perform translation with them. Training run times are reported based on Hadoop MapReduce jobs running on a cluster of 50 dedicated machines, each running 2 Map or Reduce tasks each. These results demonstrate the ability for the SAMT toolkit to scale to large resource data conditions. For each of the the three data conditions we see that training the Syntax grammar takes longer to train as well as translate with. Translation quality improvements that result from using more parallel and monolingual data are clear when comparing the 67M and 230M systems. In these experiments, we see small but consistent improvements from the introduction of SAMT labels, in line with experiments in (Zollmann et al., 2008). Overall, translation quality results reported here are competitive with reported results in the literature and constitute a valid baseline for further research.

5. Conclusions and Resources

In this paper we have described the SAMT on Hadoop toolkit, an end-to-end framework for large scale grammar based statistical machine translation. We discussed the methodology of the SAMT approach, and described important toolkit parameters that affect translation quality and run time. Built upon the open-source Hadoop distributed computation framework, our toolkit is able to scale to build grammars for large scale translation tasks in reasonable time frames. The toolkit can be easily extended to experiment with alternative grammar extraction and decoding techniques.

Bibliography

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*.

Chappelier, J.C. and M. Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of Tabulation in Parsing and Deduction (TAPD)*, pages 133–137, Paris.

Charniak, Eugene. 2000. A maximum entropy-inspired parser. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Chiang, David. 2007. Hierarchical phrase based translation. *Computational Linguistics*.

Chiang, David and Kevin Knight. 2006. An introduction to synchronous grammars. In *Tutorials at the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Cutting, Doug and Eric Baldeschwieler. 2007. Meet Hadoop. In *O'Reilly Open Software Convention*, Portland, OR.

Dean, Jeffrey and Sanjay Ghemawat. 2004. Mapreduce: Simplified data process on large cluster. In *Proceedings of Symposium on Operating System Design and Implementation*.

Doddington, George. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *In Proceedings ARPA Workshop on Human Language Technology*.

Dyer, Christopher, Aaron Cordova, Alex Mont, and Jimmy Lin. 2008. Fast, easy, and cheap: Construction of statistical machine translation models with mapreduce. In *Proceedings of the Workshop on Statistical Machine Translation, ACL*.

Galley, Michael, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*.

Huang, Liang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Koehn, Philipp, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*.

Och, Franz J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Och, Franz J. and Hermann Ney. 2003. A systematic comparison of various alignment models. *Computational Linguistics*.

Och, Franz J. and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Steedman, Mark. 1999. Alternative quantifier scope in CCG. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Stolcke, Andreas. 2002. SRILM —an extensible language modeling toolkit. In *Proceedings of the International Conferrence on Spoken Language Processing (ICSLP)*.

Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*.

Zollmann, Andreas and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation, HLT/NAACL*, New York, June.

Zollmann, Andreas, Ashish Venugopal, Franz J. Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of the Conference on Computational Linguistics (COLING)*.

Zollmann, Andreas, Ashish Venugopal, and Stephan Vogel. 2008. The CMU Syntax-Augmented Machine Translation System: SAMT on Hadoop with N-best Alignments. In *Proc. of the International Workshop on Spoken Language Translation*, pages 18–25, Hawaii, USA.