The CMU Syntax-Augmented Machine Translation System: SAMT on Hadoop with N-best alignments

Andreas Zollmann, Ashish Venugopal, Stephan Vogel

interACT, Language Technology Institute
School of Computer Science
Carnegie Mellon University, Pittsburgh, USA
{zollmann,ashishv,vogel+}@cs.cmu.edu

Abstract

We present the CMU Syntax Augmented Machine Translation System that was used in the IWSLT-08 evaluation campaign. We participated in the Full-BTEC data track for Chinese-English translation, focusing on transcript translation. For this year's evaluation, we ported the Syntax Augmented MT toolkit [1] to the Hadoop MapReduce [2] parallel processing architecture, allowing us to efficiently run experiments evaluating a novel "wider pipelines" approach to integrate evidence from N-best alignments into our translation models. We describe each step of the MapReduce pipeline as it is implemented in the open-source SAMT toolkit, and show improvements in translation quality by using N-best alignments in both hierarchical and syntax augmented translation systems.

1. Introduction

While the IWSLT evaluation represents a limited domain, scarce resource condition machine translation task, the choice of models and experimental design parameters can create a computationally expensive experimental life-cycle. For such resource-scarce machine translation tasks, we hope to squeeze every last drop of useful data out of our valuable but limited parallel corpora. Typically, machine translation systems use several early pruning strategies to keep component models small, allowing them to easily fit into memory during parameter estimation phases and when used during translation (decoding). For scare resource tracks, we prefer to retain as much data as possible, letting the decoder make the final decisions to deliver the best translation quality.

In this work, we use the IWSLT evaluation task to experiment with widening the pipeline that carries data from the initial parallel corpora, through the identification and estimation of probabilistic synchronous context-free grammar (PSCFG) rules, and finally to the decoder, where this data is used with an n-gram language model to generate translations. Current phrase-based and hierarchically structured systems rely on the output of a sequential "pipeline" of maximum *a posteriori* inference steps to identify hidden translation structure and estimate the parameters of their translation models. The first step in this pipeline typically involves learning word-alignments [3] over parallel sentence-aligned training data. The outputs of this step are the word alignment model's most probable word-to-word correspondences within each parallel sentence pair. These alignments are used as the input to a phrase extraction step, where multi-word phrase pairs are identified and scored (with multiple features) based on statistics computed across the training data. The most successful methods extract phrases that adhere to heuristic constraints [4, 5]. Thus, errors made within the single-best alignment are propagated (1) to the identification of phrases, since errors in the alignment affect which phrases are extracted, and (2) to the estimation of phrase weights, since each extracted phrase is counted as evidence for relative frequency estimates. Methods like those described in [6] and [7, 8] address this problem by jointly modeling alignment and phrase identification, yet have not achieved the same empirical results as surface heuristic based methods, or require substantially more computational effort to train.

For this evaluation we experimented with an approach that "widens" the pipeline, rather than performing two steps jointly. We present N-best alignments to the downstream phrase extraction algorithm and define a probability distribution over these alternatives to generate expected, possibly fractional counts for the extracted translation rules, under that distribution. These fractional counts are then used when assigning weights to rules. This technique is directly applicable to both flat (phrase-based) and hierarchically-structured translation models. This approach has the potential to have a significant impact on hierarchically structure models since the choice of initial phrases has an impact on the re-ordering as well as translation operations.

Since our experiments vary parameters at the start of this sequential pipeline and because we want to ensure that all extracted rules are used to estimate the features of our model, we face a significant computational challenge. Using $N = 50$ best alignments to extract rules, we can easily generate over 2GB of compressed data (text format SAMT rules) that needs to be manipulated and sorted to generate features on each rule. While this data is still tractable on current hardware, applying this wider pipeline technique to large

scale data becomes quickly unfeasible on a single machine. We take this opportunity to port the Syntax Augmented Machine Translation (SAMT) toolkit to the Hadoop MapReduce [2] parallel processing architecture. Extending the SAMT open-source toolkit has allowed us to experiment with wider pipelines for the IWSLT 2008 evaluation task as well as apply these techniques to large scale tasks such as those in the NIST MT evaluation.

In Section 2 we formally define PSCFGs and the features on each rule that are estimated from rule occurrence data. In Section 3, we detail how the SAMT pipeline, from extraction of rules through to decoding and Minimum Error Rate training [9] are ported to the Hadoop MapReduce architecture. We present runtimes for end-to-end systems built on small, medium and large resource conditions to show the effectiveness of our implementation. We then show how we used SAMT on Hadoop, extending [1] to handle N-best alignments to deliver significant improvements in translation quality for the IWSLT evaluation.

2. Synchronous Grammars for SMT

Probabilistic synchronous context-free grammars (PSCFGs) are defined by a source terminal set (source vocabulary) $\mathcal{T}_S$, a target terminal set (target vocabulary) $\mathcal{T}_T$, and a shared nonterminal set $\mathcal{N}$, and induce rules of the form

$$X \rightarrow \langle \gamma, \alpha, \sim, w \rangle$$

where

- $X \in \mathcal{N}$ is a nonterminal,
- $\gamma \in (\mathcal{N} \cup \mathcal{T}_S)^*$ is a sequence of nonterminals and source terminals,
- $\alpha \in (\mathcal{N} \cup \mathcal{T}_T)^*$ is a sequence of nonterminals and target terminals,
- the count $\#\mathrm{NT}(\gamma)$ of nonterminal tokens in γ is equal to the count $\#\mathrm{NT}(\alpha)$ of nonterminal tokens in α,
- $\sim$: $\{1, \ldots, \#\mathrm{NT}(\gamma)\} \rightarrow \{1, \ldots, \#\mathrm{NT}(\alpha)\}$ is a one-to-one mapping from nonterminal tokens in γ to nonterminal tokens in α, and
- $w \in [0, \infty)$ is a nonnegative real-valued weight assigned to the rule.

In our notation, we will assume $\sim$ to be implicitly defined by indexing the NT occurrences in γ from left to right starting with 1, and by indexing the NT occurrences in α by the indices of their corresponding counterparts in γ. Syntax-oriented PSCFG approaches often ignore source structure, focusing instead on generating syntactically well-formed target derivations. [10] uses a single nonterminal category, [11] use syntactic constituents for the PSCFG nonterminal set, and [1] take advantage of CCG-inspired "slash" categories [12] and concatenated "plus" categories.

We now briefly describe the identification and estimation of PSCFG rules from parallel sentence aligned corpora under the framework proposed by [1]. Our contribution of integrating evidence from N-best alignments can be applied to any of the other PSCFG approaches mentioned above in a straight-forward manner.

2.1. Grammar Construction

[1] describe a process to generate a PSCFG given parallel sentence pairs $\langle f, e \rangle$, a parse tree π for each e, the maximum *a posteriori* word alignment a over $\langle f, e \rangle$, and a set of phrase pairs $Phrases(a)$ identified by any alignment-driven phrase induction technique such as e.g. [4, 5].

Each phrase in $Phrases(a)$ is first annotated with a syntactic category to produce initial **rules**, where γ is set to the source side of the phrase, α is set to the target side of the phrase, and X is assigned based on the corresponding target side span in π. If the target span of the phrase does not match a constituent in π, heuristics are used to assign categories that correspond to partial rewriting of the tree. These heuristics first consider concatenation operations, forming categories like "NP+VP", and then resort to CCG style "slash" categories like "NP/NN.". The SAMT toolkit can be used to generate an SAMT grammar as well as a purely hierarchical grammar that uses a single generic nonterminal symbol $\_X$ [10]. The syntax-augmented grammar induction module also generates a purely hierarchical variant for each syntactic rule that is generated, giving the decoder the option of using labelled or non-labelled ($\_X$) rules during translation.

These initial rules form the lexical basis for generalized rules that include nonterminals in γ and α. Following the DOP-inspired [13] rule generalization technique proposed by [10], one can now generalize each **identified** rule (initial or already partially generalized)

$$N \rightarrow f_1 \ldots f_m / e_1 \ldots e_n$$

for which there is an **initial** rule

$$M \rightarrow f_i \ldots f_u / e_j \ldots e_v$$

where $1 \leq i < u \leq m$ and $1 \leq j < v \leq n$, to obtain a new rule

$$\begin{aligned} N \quad \rightarrow \quad & f_1 \ldots f_{i-1} M f_{u+1} \ldots f_m / \\ & e_1 \ldots e_{j-1} M e_{v+1} \ldots e_n \end{aligned}$$

where the two instances of M are mapped under $\sim$. The recursive form of this generalization operation allows the generation of rules with multiple nonterminal symbols. Since we only conside initial phrases up to a fixed length (10 in this work), and only allow a fixed number of nonterminals per rule (2), this operation has a runtime that is polynomial as a function of $|Phrases(a)|$.

2.2. Decoding

Given a source sentence f, the translation task under a PSCFG grammar can be expressed analogously to monolingual parsing with a CFG. We find the most likely derivation

D of the input source sentence while reading off the English translation from this derivation:

$$\hat{e} = \mathrm{tgt}\left(\underset{D:\mathrm{src}(D)=f}{\arg\max}\ p(D)\right) \quad (1)$$

where $\mathrm{tgt}(\cdot)$ maps a derivation to its target yield and $\mathrm{src}(\cdot)$ maps a derivation to its source yield.

Our 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:

$$p(D) = \frac{p_{\mathrm{LM}}(\mathrm{tgt}(D))^{\lambda_{\mathrm{LM}}} \times \prod_{r \in D} \prod_i \phi_i(r)^{\lambda_i}}{Z(\lambda)} \quad (2)$$

where ϕ_i is a feature function on rules, p_{LM} is an n-gram probability of the target yield $\mathrm{tgt}(D)$, and $Z(\lambda)$ is a normalization constant chosen such that the probabilities sum up to one.[1] The computational challenges of this search task (compounded by the integration of the language model) are addressed elsewhere [14, 15]. All feature weights λ_i are trained in concert with the language model weight λ_{LM} via minimum-error training (MER) [9]. Now, we focus on the estimation of the feature *values* ϕ during the grammar construction process. The feature values are statistics estimated from rule counts.

2.3. Feature Value Statistics

The features ϕ represent multiple criteria by which the decoding process can judge the quality of each rule and, by extension, each derivation. We include both real-valued and boolean-valued features for each rule. The following probabilistic quantities are estimated and used as feature values:

- $\hat{p}(r|\,\mathrm{lhs}(X))$: probability of a rule given its left-hand side category;
- $\hat{p}(r|\,\mathrm{src}(r))$: probability of a rule given its source side;
- $\hat{p}(r|\,\mathrm{tgt}(r))$: probability of a rule given its target side;
- $\hat{p}(\mathrm{ul}(\mathrm{tgt}(r))|\,\mathrm{ul}(\mathrm{src}(r)))$: probability of the unlabeled target side of the rule given its unlabeled source side; and
- $\hat{p}(\mathrm{ul}(\mathrm{src}(r))|\,\mathrm{ul}(\mathrm{tgt}(r)))$: probability of the unlabeled source and target side of the rule given its unlabeled target side.

In our notation, lhs returns the left-hand side of a rule, src returns the source side γ, and tgt returns the target side α of a rule r. The function ul removes all syntactic labels from its arguments, but retains ordering notation. For example, $\mathrm{ul}(\text{NP+AUX}_1\text{does not go}) = \square_1$ does not go.

The last two features represent the same kind of relative frequency estimates commonly used in phrase-based systems. The ul function allows us to calculate these estimates for rules with nonterminals as well. To estimate these probabilistic features, we use maximum likelihood estimates based on counts of the rules extracted from the training data. For example, $\hat{p}(r|lhs(r))$ is estimated by computing $\#(r)/\#(\mathrm{lhs}(r))$, aggregating counts from all extracted rules.

As in phrase-based translation model estimation, ϕ also contains two lexical weights $\hat{p}_w(\mathrm{lex}(\mathrm{src}(r))|\,\mathrm{lex}(\mathrm{tgt}(r)))$ and $\hat{p}_w(\mathrm{lex}(\mathrm{tgt}(r))|\,\mathrm{lex}(\mathrm{src}(r)))$ [4] that are based on the lexical symbols of γ and α. These weights are estimated based on a pair of statistical lexicons that represent $\hat{p}(s|t), \hat{p}(t|s)$, where s and t are single words in the source and target vocabulary. These word-level translation models are typically estimated by maximum likelihood, considering the word-to-word links from "single-best" alignments as evidence.

ϕ contains several boolean features that indicate whether: (a.) the rule is purely lexical in α and γ, (b.) the rule is purely *non-lexical* in α and γ, (c.) the ratio of lexical source and target words in the rule is between 1/5 and 5. ϕ also contains a feature that reflects the number of target lexical symbols and a feature that is 1 for each rule, allowing the decoder to prefer shorter (or longer) derivations based on the corresponding weight in λ.

3. SAMT and Hadoop

Given a cluster of machines, there exist several solutions to deploy these resources for computational work. Systems like Condor ([16]) and Sun's Grid Engine, provide coarse-grained job management (accepting, scheduling, dispatching, and managing the remote execution) to a cluster of machines. These systems are primarily responsible for managing the smooth execution of jobs submitted to the cluster, while placing minimal constraints on the nature of the running jobs.

Alternatively, the MapReduce [2] architecture is a programming model where large computational tasks are split into two distinct phases, a Map phase and a Reduce phase. In the Map phase, unstructured input data is processed by parallel tasks generating intermediate output in the form of key-value pairs. In the Reduce phase, tasks running in parallel receive this intermediate data with the guarantee that each process will receive all intermediate key-value pairs that share the same key. Under this framework, large computational tasks and task pipelines (like identifying and estimating parameters for SAMT rules and running decoding and MER training) can be distributed to run on a cluster of commodity hardware.

In order to use our existing codebase (written in C++) as much as possible, we take advantage of Hadoop Streaming which allows arbitrary executables to serve as Map and Reduce tasks taking text input on the standard input stream and outputting text on the standard output stream. The architecture treats the tab symbol as a key-value separator. Each step of SAMT pipeline is described as a MapReduce phase where outputs from each phase are written to HDFS, a highly fault

[1] Note that we never need to actually compute $Z(\lambda)$ since we are merely interested in the maximum-probability derivation.

tolerant distributed file system. The Hadoop architecture pipes data from HDFS as input to the next phase. Our implementation is based on a Hadoop-on-Demand (HOD) implementation on a Yahoo! research cluster for Carnegie Mellon University [17]. HOD allocates dedicated nodes from a large cluster to each MapReduce job, and de-allocates these nodes when the job is finished. We use Hadoop v0.17.4, and up to two tasks run on each node. Each node has 6 GB of physical memory shared across two CPUs.

For each MapReduce phase of the pipeline, we specify the MapInput (data received by the Map task), MapOptions (parameters to the Map task), MapOutput (key-value pairs output by the Map task), ReduceInput (input guaranteed to be contiguous to the Reduce task), ReduceOptions (parameters to the Reduce task), and ReduceOutput (unstructured output format from the Reduce task). The SAMT pipeline assumes input of the format $e, f, a(e, f), \pi(e)$, where e is a target language sentence from the training data, f is a source language sentence from the training data, $a(e, f)$ is a word-to-word alignment [18, 3] on e, f and $\pi(e)$ is phrase structure parse tree on e. The SAMT pipeline can be split into the following phases: Phrase Extraction, Rule Extraction, Rule Filtering, LM filtering (optional), Decoding, N-Best Merge and MER Training. In each phase we try to limit the number of key-value pairs to reduce I/O overhead, outputting multiple values that share the same key from the same Map task on a single line. The Rule Filtering and LM Filtering phases build sentence specific models for each sentence in the development and test corpus allowing the Decoding phrase to load these models directly into memory. We now describe each phase.

Phrase Extraction Map:
- MapInput: Input lines of the form $f, e, a(e, f), \pi(e)$
- MapOptions: Maximum extractable phrase length
- MapOutput: $key = sno$,
  $value = \langle f, e, Phrases(e, f), \pi(e)\rangle$ where sno is the respective sentence number

The Phrase Extraction phase identifies $Phrases(e, f)$ based on the word-aligned data and adds it to the training data stream. There is no Reduce step in this Phase.

Rule Extraction Map:
- MapInput: Each line contains $f, e, Phrases(e, f), \pi(e)$
- MapOptions: Maximum number of nonterminals per rule, maximum length of γ, options to select lhs from π
- MapOutput: key $= \mathrm{ul}(\gamma)$
  value $= \langle \gamma, \alpha, \mathrm{lhs}, 1\rangle$ and
  key $= \mathrm{ul}(\alpha)$value $= 1$

Rule Extraction uses its input to generate PSCFG rules via the procedure in Section 2.1, taking several parameters that define the grammar. MapOutput ouputs the unlabeled source side of each rule $\mathrm{ul}(\gamma)$ as key, with the rule itself as value. Since the subsequent Reduce input will see rules grouped by $\mathrm{ul}(\gamma)$, efficient computation of features $\hat{p}(r|\,\mathrm{src}(r))$, $\hat{p}(\mathrm{ul}(\mathrm{tgt}(r))|\,\mathrm{ul}(\mathrm{src}(r)))$ is possible in the Reduce step. MapOutput also outputs occurrence statistics for each lhs and for each unlabelled target side of the rule in order to compute additional features in ϕ in later phases.

Rule Extraction Reduce:
- ReduceInput: All rules that share the same $\mathrm{ul}(\gamma)$
- ReduceOptions: Minimum occurrence counts for lexical and nonlexical rules
- ReduceOutput: Rules with subset of features in ϕ. Rules that share the same $\mathrm{ul}(\gamma)$ are output on the same line.

Features $\hat{p}(r|\,\mathrm{src}(r))$, $\hat{p}(\mathrm{ul}(\mathrm{tgt}(r))|\,\mathrm{ul}(\mathrm{src}(r)))$ are computed in the Reduce step since all rules that share the same unlabelled source are available contiguously to the Reduce step. Key-value pairs indicating lhs and $\mathrm{ul}(\alpha)$ are simply accumulated.

In the Rule Filtering phase, we select those rules that can possibly be applied to each sentence in the development and test corpus in the Map step, and in the Reduce step we take these rules, add special SAMT rules to handle unknown words and glue-rule [10], resulting in a sentence specific PSCFG.

Rule Filtering Map:
- MapInput: Rules from Rule Extraction stage (single source as key with multiple rules as values)
- MapOptions: Source corpus to filter rules against (whole source corpus is loaded into memory)
- MapOutput: key $= sno$
  value $= \langle \gamma, \alpha, \phi\rangle$
  such that all words in the γ are in sentence sno in the source corpus

Count information for lhs and $\mathrm{ul}(\alpha)$ is keyed for every sentence. In the filtering step this count information is used to generate the remaining relative frequency features.

Rule Filtering Reduce:
- ReduceInput: All rules and special counts for a single sentence
- ReduceOptions: Additional models to generate the remaining features ϕ
- ReduceOutput: Rules with fully formed ϕ for a single sentence. Rules for a particular sentence are written to a canonically named file.

The Rule Filtering phase outputs canonical per-sentence grammar files as a side-effect file on HDFS, rather than on the standard output stream. On the standard output stream we output the potential target vocabulary for each sentence based on the sentence-specific grammar.

LM Filtering is an optional phase to run with the n-gram language models used for decoding are too large to fit in memory. By using the potential target language vocabulary

System Name	Words in Target Text	# LM n-grams	Dev. Set	Test Set 1	Test Set 2
IWSLT	632K	431,292 ($n = 5$)	IWSLT Devset4	IWSLT07	IWSLT08
67M	67M	102,924,025 ($n = 4$)	MT05	MT06	n/a
230M	230M	273,233,010 ($n = 5$)	MT05	MT06	n/a

Table 1: Data configurations used to evaluated SAMT on Hadoop. The number of words in the target text and the number of language model 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	End-to-End Time	Dev. BLEU	Test-1 BLEU	Test-2 BLEU
IWSLT Hier.	12 mins	0.278	0.360	0.427
IWSLT Hier. with full-sentence rules		0.277	0.367	0.460
IWSLT Syntax	26 mins	0.296	0.335	0.430
IWSLT Syntax with full-s. rules		0.301	0.361	0.440
67M hier	1 hrs 10 mins	0.345 (lc)	0.323 (lc)	n/a
230M hier	7 hrs	0.362 (lc)	0.337 (lc)	n/a

Table 2: Translation quality as measured by IBM-BLEU on each resource track for appropriate evaluation data sets. (lc) indicates that scores are lower-cased, otherwise scores are mixed case 4-gram BLEU

for each sentence, we can build sentence-specific n-gram language models which are much smaller without losing any relevant parameters.

LM Filtering Map:

- MapInput: Each line is a line from an ARPA format LM
- MapOptions: Access to a $sno \rightarrow vocabulary$ map from the filtering stage (loaded into memory)
- MapOutput: key $= sno$ value $= t_1 \cdots t_n$ if every t_i is in the target vocabulary of sno.

The Map step selects relevant n-gram lines for each sentence, while the Reduce step re-builds a valid ARPA LM. Just like the Rule Filtering phase, LM Filtering produces canonically named sentence-specific language model files as side-effects on HDFS.

LM Filtering Reduce:

- ReduceInput: All n-grams that are compliant with a single sentence's vocabulary
- ReduceOuput: Statistics over n-grams are computed and output as a header to form a complete ARPA LM

The Decoding phase runs translation accessing the sentence specific translation and language models and outputting an n-best list for each sentence.

Decoding Map:

- MapInput: A single sentence to translate per line with sno information
- MapOptions: Options typically passed to a decoder to run translation. We also specify a path to a HDFS directory containing per-sentence grammars and language models.
- MapOutput: key $= sno$ value $=$ n-best list

If we are running Minimum Error Rate (MER) training, we perform an additional Merge phase that takes n-best list output from all MER iterations and performs a trivial MapReduce to merge n-best lists across iterations and remove duplicates. Minimum Error Rate training is implemented as a MapReduce task as well. We do not parallelize the inner-working of the MER process, rather we simply allow multiple initial parameter configurations to be evaluated in parallel. In order to pass *different* parameters to each MER task, we define MER MapReduce as follows:

MER Map:

- MapInput: N-Best lists for MER optimization
- MapOptions: Multiple parameter conditions for MER. Each parameter condition includes initial parameters to start MER
- MapOutput: $key =$ one MER parameter config. $value =$ all optimization data

In the Reduce step, each Reduce task receives all the N-best list data *and* parameters to run the optimization with, allowing each Reducer to run optimization with different parameters.

3.1. SAMT on MapReduce results

To demonstrate the effectiveness of our implementation of SAMT on Hadoop, we test our system on three resource conditions representing the full spectrum of translation scenarios. These conditions are described in Table 1.

The 67M and 230M resource conditions represent medium and large data task both in terms of available parallel data and monolingual LM data. Training and evaluation corpora for these resource conditions are available via LDC for the NIST MT evaluation task. In Table 2, we report approximate wall-clock times for end-to-end systems built on

these data sets with corresponding BLEU scores for development and test data. Timing is measured from Phrase Extraction phase until Decoding phase using a 20 node HOD cluster. We report results for IWSLT hierarchical and syntax augmented systems, and purely hierarchical systems for the larger corpora.

For the IWSLT data track, we also tested variants of our hierarchical and syntax-augmented systems, which treat the beginning-of-sentence and end-of-sentence as regular words, thereby allowing for extraction of rules such as (in the syntax-augmented case): `S -> <s> NP VP . </s> / <s> VP NP . </s>`

To limit the number of such rules, we only allow the ones spanning the full sentence (i.e., including $< s >$ as well as $< /s >$). As can be seen from Table 2, while not having much impact on development-set performance, adding these full-sentence results in impressive increases in BLEU score on the test sets.

The results in Table 2 demonstrate that we can build high quality translations systems using the SAMT toolkit on Hadoop within reasonable time frames. We now use the toolkit to experiment with wider pipelines, integrating N-best alignment evidence.

4. N-best Evidence

The PSCFG rule extraction procedure described above relies on high quality word alignments and parses. The quality of the alignments affects the set of phrases that can be identified by the heuristics in [4]. Improving or diversifying the set of initial phrases also affects the rules with nonterminals that are identified via the procedure described above. Since PSCFG systems rely on rules with nonterminal symbols to represent reordering operations, the set of these initial phrases can have a profound impact on translation quality. Several recent studies [19, 20, 21, 22], explore the relationship between the quality of the initial decisions in the "pipeline" and final translation quality.

Here we experiment with weighted N-best alignments to build PSCFGs and estimate their features. Our approach toward the integration of N-best evidence into the grammar construction process allows us to take advantage of the diversity found in the N best alternatives, while reducing the negative impact of errors made in these alternatives.

4.1. Counting from N-Best Lists

For this evaluation we experimented with the extraction of PSCFG rules from N-best alignments making use of probability distributions over these alternatives to assign fractional posterior counts to each extracted rule.

We follow [4] in generating a refined bidirectional alignment using the heuristic algorithm "grow-diag-final-and" described in that work. Since we require N-best alignments, we first extract N-best alignments in each direction, and then perform the refinement technique to all N^2 bidirectional alignment pairs. The resulting alignments are assigned the probability $(p_f \times p_r)^\alpha$ where p_f is the candidate probability for the forward alignment and p_r is the candidate probability to the reverse alignment.

The parameter α controls the entropy of the resulting distribution over candidate alignments. The value $\alpha = 1$ corresponds to the joint alignment distribution under assumption of independence of forward and reverse distributions. Higher values of $\alpha > 1$ make the distribution more peaked (affecting the estimation of features on rules from these alignments), while values of $0 \leq \alpha < 1$ make the distribution more uniform. A more peaked distribution favors rules from the top alignments, while a more uniform one gives rules from lower performing alignments more of a chance to participate in translation. As a special case, $\alpha = 1/2$ effectively uses the geometric mean of forward and reverse alignment distributions.

Taking the alignment N-best list to define a posterior distribution over alignments we can estimate the posterior probability of each rule that might be extracted for each alignment pair. Given the alignments $a_1, ..., a_N$, with probabilities $p(a_1 \mid e, f), ..., p(a_N \mid e, f)$ estimated as described above, we approximate the alignment posterior by re-normalizing:

$$\hat{p}(a_i) = p(a_i \mid e, f) \Big/ \sum_{j=1}^{N} p(a_j \mid e, f) \tag{3}$$

Given a word-alignment, we can extract rules as described in Section 2.1. Our approach is to extract rules from each of the $\{a_1, ..., a_N\}$ alignments, incrementing the partial count of each rule extracted by $\hat{p}(a_i)$. A rule r's total count for the sentence pair $\langle f, e \rangle$ is thus:

$$\sum_{i=1}^{N} \hat{p}(a_i) \cdot \begin{cases} 1 & \text{if } r \text{ can be extracted from} \\ & \quad e, f, a_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In practice, this can be computed more efficiently through structure-sharing. Note that if $N = 1$, this counting method corresponds to the original counting method.

Instead of using the simple counts for rules given the derivation inferred using the maximum *a posteriori* estimated alignment, we now use the expected counts under the approximate posterior. These posteriors encode (in a principled way) a measurement of confidence in substructures used to generate each rule. Possible rule instances supported by more and by more likely alignments should, intuitively, receive higher counts (approaching 1 as certainty increases, supported by more and higher-probability alternatives), while rule instances that rely on low probability or fewer alignments will get lower counts (approaching 0 as uncertainty increases). In practice however, we find that the choice of α had no consistent effect on the IWSLT evaluation sets—in fact, for $N \leq 10$, even $\alpha = 0$, corresponding to the uniform alignment distribution, did not harm development and test set BLEU scores. More experiments varying α

System	# Rules (1 sent.)	Dev	2007	2008	2007 Time (s)	2008 Time (s)
Syntax $N = 1$ (lex=1st)	400K	0.309	0.355	0.453	8108	8367
Syntax $N = 1$	420K	0.301	0.361	0.440	8024	8250
Syntax $N = 5$	680K	0.322	0.374	0.470	15376	15577
Syntax $N = 10$	900K	0.313	0.382	0.467	19298	19469
Syntax $N = 50$	1500K	0.316	0.370	0.478	29500	30894
Hier $N = 1$	10K	0.277	0.367	0.460	895	1451
Hier $N = 5$	12K	0.286	0.374	0.472	906	1476
Hier $N = 10$	13K	0.291	0.382	0.477	944	1516
Hier $N = 50$	14K	0.282	0.384	0.463	979	1596

Table 3: Grammar statistics and translation quality (IBM-BLEU) on development (IWSLT Devset4) and test sets (IWSLT 2007, 2008) when integrating N-best alignments. # Rules reflect rules that are applicable to the first sentence in IWSLT 2007. Decoding times in seconds are cumulative over all sentences in respective test set.

and additional details regarding selecting N-best alignments and even N'-best parses can be found in [23]. We use $\alpha = 1$ in the experiments presented here.

5. Translation Results

5.1. Experimental Setup

We present results on the IWSLT 2007 and 2008 Chinese-to-English translation task, based on the full BTEC corpus of travel expressions with 120K parallel sentences (906K source words and 1.2M target words) as well as the evaluation corpora from the evaluation years preceding 2007. The development data consists of 489 sentences (average length of 10.6 words) from the 2006 evaluation, the 2007 test set contains 489 sentence (average length of 6.47 words) sentences and the 2008 test set contains 507 sentences (average length of 5.59 words). Word alignment was trained using the GIZA++ toolkit.

Initial phrases of up to length 10 were identified using the heuristics proposed by [4]. Rules with up to 2 nonterminals are extracted using the SAMT toolkit [1], modified to handle N-best alignments and parses and posterior counting. Note that lexical weights [4] as described above are assigned to ϕ based on "single-best" word alignments. Rules that receive zero probability value for their lexical weights are immediately discarded, since they would then have a prohibitively high cost when used during translation. Rules extracted from single-best evidence as well as N best evidence can be discarded in this way. As an alternative, we use IBM Model 4 translation weights to estimate lexical weights. Using these IBM Model 4 weights allows a larger number of rules to be added to the grammar since more rules have non-zero lexical weights.

The n-gram language model is trained on the target side of the parallel training corpus and translation experiments use the decoder and MER trainer available in the same toolkit. We use the cube-pruning decoder option [14] in these experiments.

5.2. Empirical Results

We measure translation quality using the mixed-cased IBM-BLEU [24] metric as we vary N. Each value of N implies that the first N alignment alternatives have been considered when building the grammar. For each grammar we also track the number of rules relevant for the first sentence in the IWSLT 2007 test set (grammars are subsampled on a per-sentence basis to keep memory requirements low during decoding). We also note the number of seconds required to translate each test set.

N-best alignments (Syntax augmented grammar. Table 3 shows translation results on the IWSLT translation task for the development (IWSLT 2006) and two test corpora (IWSLT 2007 and 2008) using the Syntax Augmented and Hierarchical grammars. In this table we vary the number of alternative alignments, consider first-best (1), 5, 10 and 50 best alternatives. We also compare using lexical weights from the first-best alignment ($lex = 1st$) and the default Model 4 weights.

For the Syntax-Augmented grammar, using Model 4 weights slightly increases the number of rules in the grammar, but only adds benefit for the 2007 test set. We continue to use Model 4 weights for the remaining experiments since we do not want to discard rules based on the lexical weights. Increasing $N = 1$ to $N = 5$ brings significant improvements in translation quality on all 3 evaluation corpora, while increasing N further to $N = 10$ and $N = 50$ retains the improvements, but at the cost of a significantly larger grammar and decoding times.

N-best alignments (Hierarchical grammar). Similar results are seen with the hierarchical grammar. We see clear improvements when moving to $N = 5$, and even further small improvement up to $N = 10$, but a slight degradation going further to $N = 50$. Surprisingly, while scores on the development set are significantly lower with the purely hierarchical grammar compared to the Syntax Augmented grammar, unseen test set scores are very similar, and achieved at significantly lower decoding times. Since the number of features in ϕ are very similar for both models, it is unlikely that this discrepancy is solely due to overfitting during MER

training. It is more likely that the overfitting is due to the number of rules used in the Syntax Augmented grammar being one to two orders of magnitude more than the Hierarchical's. Even though the rules are not free parameters, overfitting is expected to occur if the development set is more similar to the training corpus than the test set(s). (Judging from the sentence length distribution alone, this seems indeed to be the case.)

6. Conclusion

For the 2008 IWSLT evaluation we evaluated the use of N-best alignments to take more advantage of the available data in this scarce resource scenario task. While we did see substantial improvements in translation quality using N-best alignments, we note that the variance in BLEU score results on the IWSLT evaluation sets is still very high. While none of the result presented here can be considered "statistically significant" according to the testing criteria in [25], we are able to see trends across multiple evaluation corpora that we can use to evaluate the potential of a technique. In order to efficiently run these experiments, which modify the very beginning of the long MT pipeline, we ported the SAMT toolkit to the Hadoop parallel processing architecture, allowing us to quickly run experiments for this evaluation as well as for medium and large data scenarios. This Hadoop port is integrated into the publicly available open-source SAMT toolkit.

7. Acknowledgments

This work would not have been possible without access to the M45 cluster, which was generously granted by Yahoo!. Our research was in part supported by DARPA under contract HR0011-06-2-0001 (GALE).

8. References

[1] A. Zollmann and A. Venugopal, "Syntax augmented machine translation via chart parsing," in *Proc. of the Workshop on SMT, HLT/NAACL*, New York, June 2006.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proc. of Symposium on Operating System Design and Implementation*, 2004.

[3] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: parameter estimation," *Computational Linguistics*, 1993.

[4] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *Proc. of HLT/NAACL*, 2003.

[5] F. J. Och and H. Ney, "The alignment template approach to statistical machine translation," *Computational Linguistics*, 2004.

[6] D. Wu, "Stochastic inversion transduction grammars and bilingual parsing of parallel corpora," *Computational Linguistics*, 1997.

[7] D. Marcu and W. Wong, "A phrase-based, joint probability model for statistical machine translation," in *Proc. of EMNLP*, 2002.

[8] J. DeNero, D. Gillick, J. Zhang, and D. Klein, "Why generative phrase models underperform surface heuristics," in *Proc. of the Workshop on SMT, ACL*, 2006.

[9] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proc. of the ACL*, 2003.

[10] D. Chiang, "A hierarchical phrase-based model for statistical machine translation," in *Proc. of the Annual Meeting of the ACL*, 2005.

[11] M. Galley, M. Hopkins, K. Knight, and D. Marcu, "Scalable inferences and training of context-rich syntax translation models," in *Proc. of HLT/NAACL*, 2006.

[12] M. Steedman, *The Syntactic Process*. MIT Press, 2000.

[13] R. Scha, "Taaltheorie en taaltechnologie; competence en performance (language theory and language technology; competence and performance)," *Computertoepassingen in de Neerlandistiek*, 1990.

[14] D. Chiang, "Hierarchical phrase based translation," *Computational Linguistics*, 2007.

[15] A. Venugopal, A. Zollmann, and S. Vogel, "An efficient two-pass approach to synchronous-CFG driven statistical MT," in *Proc. of HLT/NAACL*, 2007.

[16] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience." *Concurrency - Practice and Experience*, 2005.

[17] D. Cutting and E. Baldeschwieler, "Meet Hadoop," in *O'Reilly Open Software Convention*, Portland, OR, 2007.

[18] C. Dyer, A. Cordova, A. Mont, and J. Lin, "Fast, easy, and cheap: Construction of statistical machine translation models with mapreduce," in *Proc. of the Workshop on SMT, ACL*, 2008.

[19] C. Dyer, S. Muresan, and P. Resnik, "Generalizing word lattice translation," in *Proc. of the ACL*, 2008.

[20] H. Mi and L. Huang, "Forest-based translation rule extraction," in *Proc. of EMNLP*, 2008.

[21] K. Ganchev, J. V. Graca, and B. Taskar, "Better alignments = better translations?" in *Proc. of the ACL*, 2008.

[22] Y.-Z. Xue, S. Li, T.-J. Zhao, M.-Y. Yang, and J. Li, "Bilingual phrase extraction from n-best alignments," in *Proc. of ICICIC*, 2006.

[23] A. Venugopal, A. Zollmann, N. A. Smith, and S. Vogel, "Wider pipelines: N-best alignments and parses in MT training," in *Proc. of AMTA*, 2008.

[24] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation." in *Proc. of the ACL*, 2002.

[25] Y. Zhang and S. Vogel, "Measuring confidence intervals for the machine translation evaluation metrics," in *Proc. of TMII*, October 2004.