Akamon: An Open Source Toolkit for Tree/Forest-Based Statistical Machine Translation∗

Xianchao Wu†, Takuya Matsuzaki∗, Jun’ichi Tsujii‡
† Baidu Inc.
‡ National Institute of Informatics
‡ Microsoft Research Asia
wuxianchao@gmail.com, takuya-matsuzaki@nii.ac.jp, jtsujii@microsoft.com

Abstract

We describe Akamon, an open source toolkit for tree and forest-based statistical machine translation (Liu et al., 2006; Mi et al., 2008; Mi and Huang, 2008). Akamon implements all of the algorithms required for tree/forest-to-string decoding using tree-to-string translation rules: multiple-thread forest-based decoding, n-gram language model integration, beam- and cube-pruning, k-best hypotheses extraction, and minimum error rate training. In terms of tree-to-string translation rule extraction, the toolkit implements the traditional maximum likelihood algorithm using PCFG trees (Galley et al., 2004) and HPSG trees/forests (Wu et al., 2010).

1 Introduction

Syntax-based statistical machine translation (SMT) systems have achieved promising improvements in recent years. Depending on the type of input, the systems are divided into two categories: string-based systems whose input is a string to be simultaneously parsed and translated by a synchronous grammar (Wu, 1997; Chiang, 2005; Xiong et al., 2006), and tree/forest-string systems whose input is already a tree or a packed forest to be directly converted into a target tree or string (Ding and Palmer, 2005; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Mi et al., 2008; Mi and Huang, 2008; Zhang et al., 2009; Wu et al., 2011a, 2010; Wu et al., 2011a).

Depending on whether or not parsers are explicitly used for obtaining linguistically annotated data during training, the systems are also divided into two categories: formally syntax-based systems that do not use additional parsers (Wu, 1997; Chiang, 2005; Xiong et al., 2006), and linguistically syntax-based systems that use PCFG parsers (Liu et al., 2006; Huang et al., 2006; Galley et al., 2006; Mi et al., 2008; Mi and Huang, 2008; Zhang et al., 2009), HPSG parsers (Wu et al., 2010; Wu et al., 2011a), or dependency parsers (Ding and Palmer, 2005; Quirk et al., 2005; Shen et al., 2008). A classification1 of syntax-based SMT systems is shown in Table 1.

Translation rules can be extracted from aligned string-string (Chiang, 2005), tree-tree (Ding and Palmer, 2005) and tree/forest-string (Galley et al., 2004; Mi and Huang, 2008; Wu et al., 2011a) data structures. Leveraging structural and linguistic information from parse trees/forests, the latter two structures are believed to be better than their string-string counterparts in handling non-local reordering, and have achieved promising translation results. Moreover, the tree/forest-string structure is more widely used than the tree-tree structure, presumably because using two parsers on the source and target languages is subject to more problems than making use of a parser on one language, such as the shortage of high precision/recall parsers for languages other than English, compound parse error rates, and inconsistency of errors. In Table 1, note that tree-to-string rules are generic and applicable to many syntax-based models such as tree/forest-to-

1This classification is inspired by and extends the Table 1 in (Mi and Huang, 2008).
string models and string-to-tree model.

However, few tree/forest-to-string systems have been made open source and this makes it difficult and time-consuming to testify and follow existing proposals involved in recently published papers. The Akamon system\(^2\), written in Java and following the tree/forest-to-string research direction, implements all of the algorithms for both tree-to-string translation rule extraction (Galley et al., 2004; Mi and Huang, 2008; Wu et al., 2010; Wu et al., 2011a) and tree/forest-based decoding (Liu et al., 2006; Mi et al., 2008). We hope this system will help related researchers to catch up with the achievements of tree/forest-based translations in the past several years without re-implementing the systems or general algorithms from scratch.

## 2 Akamon Toolkit Features

Limited by the successful parsing rate and coverage of linguistic phrases, Akamon currently achieves comparable translation accuracies compared with the most frequently used SMT baseline system, Moses (Koehn et al., 2007). Table 2 shows the automatic translation accuracies (case-sensitive) of Akamon and Moses. Besides BLEU and NIST score, we further list RIBES score\(^3\), i.e., the software implementation of Normalized Kendall’s \(\tau\) as proposed by (Isozaki et al., 2010a) to automatically evaluate the translation between distant language pairs based on rank correlation coefficients and significantly penalizes word order mistakes.

In this table, Akamon-Forest differs from Akamon-Comb by using different configurations: Akamon-Forest used only 2/3 of the total training data (limited by the experiment environments and time). Akamon-Comb represents the system combination result by combining Akamon-Forest and other phrase-based SMT systems, which made use of pre-ordering methods of head finalization as described in (Isozaki et al., 2010b) and used the total 3 million training data. The detail of the pre-ordering approach and the combination method can be found in (Sudoh et al., 2011) and (Duh et al., 2011).

Also, Moses (hierarchical) stands for the hierarchical phrase-based SMT system and Moses (phrase) stands for the flat phrase-based SMT system. For intuitive comparison (note that the result achieved by Google is only for reference and not a comparison, since it uses a different and unknown training data) and following (Goto et al., 2011), the scores achieved by using the Google online translation system\(^4\) are also listed in this table.

Here is a brief description of Akamon’s main features:

- **multiple-thread** forest-based decoding: Akamon first loads the development (with source and reference sentences) or test (with source sentences only) file into memory and then perform parameter tuning or decoding in a parallel way. The forest-based decoding algorithm is alike that described in (Mi et al., 2008),

<table>
<thead>
<tr>
<th>Source-to-target</th>
<th>Examples (partial)</th>
<th>Decoding</th>
<th>Rules</th>
<th>Parser</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree-to-tree</td>
<td>(Ding and Palmer, 2005)</td>
<td>↓</td>
<td>dep.-to-dep.</td>
<td>DG</td>
</tr>
<tr>
<td>forest-to-tree</td>
<td>(Liu et al., 2009a)</td>
<td>↓ ↑↓</td>
<td>tree-to-tree</td>
<td>PCFG</td>
</tr>
<tr>
<td>tree-to-string</td>
<td>(Liu et al., 2006)</td>
<td>↑</td>
<td>tree-to-tree</td>
<td>PCFG</td>
</tr>
<tr>
<td>forest-to-string</td>
<td>(Quirk et al., 2005)</td>
<td>↑</td>
<td>dep.-to-string</td>
<td>DG</td>
</tr>
<tr>
<td></td>
<td>(Mi et al., 2008)</td>
<td>↓ ↑↓</td>
<td>tree-to-string</td>
<td>PCFG</td>
</tr>
<tr>
<td></td>
<td>(Wu et al., 2011a)</td>
<td>↑</td>
<td>tree-to-string</td>
<td>HPSG</td>
</tr>
<tr>
<td>string-to-tree</td>
<td>(Galley et al., 2006)</td>
<td>CKY</td>
<td>tree-to-string</td>
<td>PCFG</td>
</tr>
<tr>
<td>string-to-string</td>
<td>(Shen et al., 2008)</td>
<td>CKY</td>
<td>string-to-dep.</td>
<td>DG</td>
</tr>
<tr>
<td></td>
<td>(Chiang, 2005)</td>
<td>CKY</td>
<td>string-to-string</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>(Xiong et al., 2006)</td>
<td>CKY</td>
<td>string-to-string</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 1: A classification of syntax-based SMT systems. Tree/forest-based and string-based systems are split by a line. All the systems listed here are linguistically syntax-based except the last two (Chiang, 2005) and (Xiong et al., 2006), which are formally syntax-based. DG stands for dependency (abbreviated as dep.) grammar. ↓ and ↑ denote top-down and bottom-up traversals of a source tree/forest.

\(^2\)Code available at https://sites.google.com/site/xianchaowu2012

\(^3\)Code available at http://www.kecl.ntt.co.jp/icl/lirg/ribes

\(^4\)http://translate.google.com/
<table>
<thead>
<tr>
<th>Systems</th>
<th>BLEU</th>
<th>NIST</th>
<th>RIBES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google online</td>
<td>0.2546</td>
<td>6.830</td>
<td>0.6991</td>
</tr>
<tr>
<td>Moses (hierarchical)</td>
<td>0.3166</td>
<td>7.795</td>
<td>0.7200</td>
</tr>
<tr>
<td>Moses (phrase)</td>
<td>0.3190</td>
<td>7.881</td>
<td>0.7068</td>
</tr>
<tr>
<td>Moses (phrase)*</td>
<td>0.2773</td>
<td>6.905</td>
<td>0.6619</td>
</tr>
<tr>
<td>Akamon-Forest*</td>
<td>0.2799</td>
<td>7.258</td>
<td>0.6861</td>
</tr>
<tr>
<td>Akamon-Comb</td>
<td>0.3948</td>
<td>8.713</td>
<td>0.7813</td>
</tr>
</tbody>
</table>

Table 2: Translation accuracies of Akamon and the baseline systems on the NTCIR-9 English-to-Japanese translation task (Wu et al., 2011b). * stands for only using 2 million parallel sentences of the total 3 million data. Here, HPSG forests were used in Akamon.

i.e., first construct a translation forest by applying the tree-to-string translation rules to the original parsing forest of the source sentence, and then collect \( k \)-best hypotheses for the root node(s) of the translation forest using Algorithm 2 or Algorithm 3 as described in (Huang and Chiang, 2005). Later, the \( k \)-best hypotheses are used both for parameter tuning on additional development set(s) and for final optimal translation result extracting.

- language models: Akamon can make use of one or many \( n \)-gram language models trained by using SRILM\(^5\) (Stolcke, 2002) or the Berkeley language model toolkit, berkeleylm-1.0b\(^6\) (Pauls and Klein, 2011). The weights of multiple language models are tuned under minimum error rate training (MERT) (Och, 2003).

- pruning: traditional beam-pruning and cube-pruning (Chiang, 2007) techniques are incorporated in Akamon to make decoding feasible for large-scale rule sets. Before decoding, we also perform the marginal probability-based inside-outside algorithm based pruning (Mi et al., 2008) on the original parsing forest to control the decoding time.

- MERT: Akamon has its own MERT module which optimizes weights of the features so as to maximize some automatic evaluation metric, such as BLEU (Papineni et al., 2002), on a development set.

\(^5\)http://www.speech.sri.com/projects/srilm/
\(^6\)http://code.google.com/p/berkeleylm/

Figure 1: Training and tuning process of the Akamon system. Here, \( e \) = source English language, \( f \) = target foreign language.

- translation rule extraction: as former mentioned, we extract tree-to-string translation rules for Akamon. In particular, we implemented the GHKM algorithm as proposed by Galley et al. (2004) from word-aligned tree-string pairs. In addition, we also implemented the algorithms proposed by Mi and Huang (2008) and Wu et al. (2010) for extracting rules from word-aligned PCFG/HPSG forest-string pairs.

3 Training and Decoding Frameworks

Figure 1 shows the training and tuning progress of the Akamon system. Given original bilingual parallel corpora, we first tokenize and lowercase the source and target sentences (e.g., word segmentation of Chinese and Japanese, punctuation segmentation of English).

The pre-processed monolingual sentences will be used by SRILM (Stolcke, 2002) or BerkeleyLM (Pauls and Klein, 2011) to train a \( n \)-gram language model. In addition, we filter out too long sentences...
here, i.e., only relatively short sentence pairs will be used to train word alignments. Then, we can use GIZA++ (Och and Ney, 2003) and symmetric strategies, such as grow-diag-final (Koehn et al., 2007), on the tokenized parallel corpus to obtain a word-aligned parallel corpus.

The source sentence and its packed forest, the target sentence, and the word alignment are used for tree-to-string translation rule extraction. Since a 1-best tree is a special case of a packed forest, we will focus on using the term ‘forest’ in the continuing discussion. Then, taking the target language model, the rule set, and the preprocessed development set as inputs, we perform MERT on the decoder to tune the weights of the features.

The Akamon forest-to-string system includes the decoding algorithm and the rule extraction algorithm described in (Mi et al., 2008; Mi and Huang, 2008).

### 4 Using Deep Syntactic Structures

In Akamon, we support the usage of deep syntactic structures for obtaining fine-grained translation rules as described in our former work (Wu et al., 2010). Similarly, Enju, a state-of-the-art and freely available HPSG parser for English, can be used to generate packed parse forests for source sentences. Deep syntactic structures are included in the HPSG trees/forests, which includes a fine-grained description of the syntactic property and a semantic representation of the sentence. We extract fine-grained rules from aligned HPSG forest-string pairs and use them in the forest-to-string decoder. The detailed algorithms can be found in (Wu et al., 2010; Wu et al., 2011a). Note that, in Akamon, we also provide the codes for generating HPSG forests from Enju.

Head-driven phrase structure grammar (HPSG) is a lexicalist grammar framework. In HPSG, linguistic entities such as words and phrases are represented by a data structure called a *sign*. A sign gives a factored representation of the syntactic features of a word/phrase, as well as a representation of their semantic content. Phrases and words represented by signs are composed into larger phrases by applications of *schemata*. The semantic representation of the new phrase is calculated at the same time. As such, an HPSG parse tree/forest can be considered as a tree/forest of signs (c.f. the HPSG forest in Figure 2 in (Wu et al., 2010)).

An HPSG parse tree/forest has two attractive properties as a representation of a source sentence in syntax-based SMT. First, we can carefully control the condition of the application of a translation rule by exploiting the fine-grained syntactic description in the source parse tree/forest, as well as those in the translation rules. Second, we can identify sub-trees in a parse tree/forest that correspond to basic units of the semantics, namely sub-trees covering a predicate and its arguments, by using the semantic representation given in the signs. Extraction of translation rules based on such semantically-connected sub-trees is expected to give a compact and effective set of translation rules.

A sign in the HPSG tree/forest is represented by a typed feature structure (TFS) (Carpenter, 1992). A TFS is a directed-acyclic graph (DAG) wherein the edges are labeled with feature names and the nodes

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT</td>
<td>phrasal category</td>
</tr>
<tr>
<td>XCAT</td>
<td>fine-grained phrasal category</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>name of the schema applied in the node</td>
</tr>
<tr>
<td>HEAD</td>
<td>pointer to the head daughter</td>
</tr>
<tr>
<td>SEM,HEAD</td>
<td>pointer to the semantic head daughter</td>
</tr>
<tr>
<td>CAT</td>
<td>syntactic category</td>
</tr>
<tr>
<td>POS</td>
<td>Penn Treebank-style part-of-speech tag</td>
</tr>
<tr>
<td>BASE</td>
<td>base form</td>
</tr>
<tr>
<td>TENSE</td>
<td>tense of a verb (past, present, untensed)</td>
</tr>
<tr>
<td>ASPECT</td>
<td>aspect of a verb (none, perfect, progressive, perfect-progressive)</td>
</tr>
<tr>
<td>VOICE</td>
<td>voice of a verb (passive, active)</td>
</tr>
<tr>
<td>AUX</td>
<td>auxiliary verb or not (minus, modal, have, be, do, to, copular)</td>
</tr>
<tr>
<td>LEXENTRY</td>
<td>lexical entry, with supertags embedded</td>
</tr>
<tr>
<td>PRED</td>
<td>type of a predicate</td>
</tr>
<tr>
<td>ARG(x)</td>
<td>pointer to semantic arguments, x = 1..4</td>
</tr>
</tbody>
</table>

Table 3: Syntactic/semantic features extracted from HPSG signs that are included in the output of Enju. Features in phrasal nodes (top) and lexical nodes (bottom) are listed separately.
(feature values) are typed. In the original HPSG formalism, the types are defined in a hierarchy and the DAG can have arbitrary shape (e.g., it can be of any depth). We however use a simplified form of TFS, for simplicity of the algorithms. In the simplified form, a TFS is converted to a (flat) set of pairs of feature names and their values. Table 3 lists the features used in our system, which are a subset of those in the original output from Enju.

In the Enju English HPSG grammar (Miyao et al., 2003) used in our system, the semantic content of a sentence/phrase is represented by a predicate-argument structure (PAS). Figure 2 shows the PAS of a simple sentence, “John killed Mary”, and a more complex PAS for another sentence, “She ignored the fact that I wanted to dispute”, which is adopted from (Miyao et al., 2003). In an HPSG tree/forest, each leaf node generally introduces a predicate, which is represented by the pair of LEXENTRY (lexical entry) feature and PRED (predicate type) feature. The arguments of a predicate are designated by the pointers from the ARG⟨x⟩ features in a leaf node to non-terminal nodes. Consequently, Akamon includes the algorithm for extracting compact composed rules from these PASs which further lead to a significant fast tree-to-string decoder. This is because it is not necessary to exhaustively generate the subtrees for all the tree nodes for rule matching any more. Limited by space, we suggest the readers to refer to our former work (Wu et al., 2010; Wu et al., 2011a) for the experimental results, including the training and decoding time using standard English-to-Japanese corpora, by using deep syntactic structures.

5 Content of the Demonstration

In the demonstration, we would like to provide a brief tutorial on:

- describing the format of the packed forest for a source sentence,
- the training script on translation rule extraction,
- the MERT script on feature weight tuning on a development set, and,
- the decoding script on a test set.

Based on Akamon, there are a lot of interesting directions left to be updated in a relatively fast way in the near future, such as:

- integrate target dependency structures, especially target dependency language models, as proposed by Mi and Liu (2010),
- better pruning strategies for the input packed forest before decoding,
- derivation-based combination of using other types of translation rules in one decoder, as proposed by Liu et al. (2009b), and
- taking other evaluation metrics as the optimal objective for MERT, such as NIST score, RIBES score (Isozaki et al., 2010a).

Acknowledgments

We thank Yusuke Miyao and Naoaki Okazaki for their invaluable help and the anonymous reviewers for their comments and suggestions.

References


Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In Proceedings of 7th AMTA.


