PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 93 JANUARY 2010 157–166

Hierarchical Phrase-Based Grammar Extraction in Joshua

Suffix Arrays and Prefix Trees

Lane Schwartz, Chris Callison-Burch

Abstract

While example-based machine translation has long used corpus information at run-time, statistical phrase-based approaches typically include a preprocessing stage where an aligned parallel corpus is split into phrases, and parameter values are calculated for each phrase using simple relative frequency estimates. This paper describes an open source implementation of the crucial algorithms presented in (Lopez, 2008) which allow direct run-time calculation of SCFG translation rules in Joshua.

1. Introduction

A significant amount of the recent research in statistical machine translation has focused on modeling translation based on contiguous strings of words, called *phrases*, in the source language and corresponding phrases in the target language. Phrase-based translation (Och, Tillmann, and Ney, 1999, Koehn, Och, and Marcu, 2003, Marcu and Wong, 2002, Och and Ney, 2004) have proved to be very successful, and many state-of-the-art machine translation systems are based on these approaches.

A critical component in phrase-based translation is the estimation of a translation model from a word-aligned parallel text. A phrase table containing the source phrases, their target translations and their associated probabilities that is typically extracted in a preprocessing stage before decoding a test set (Koehn, Och, and Marcu, 2003, Kumar, Deng, and Byrne, 2006). An example of this preprocessing approach is found in the training scripts provided as part of the open source phrase-based Moses toolkit (Koehn et al., 2007). Hierarchical phrase-based translation (Chiang, 2005) extends phrase-based translation by allowing phrases with gaps, modeled as a synchronous context-free grammar (SCFG). The original Hiero implementation (Chiang,

Please cite this article as: Lane Schwartz, Chris Callison-Burch, Hierarchical Phrase-Based Grammar Extraction in Joshua: Suffix Arrays and Prefix Trees. The Prague Bulletin of Mathematical Linguistics No. 93, 2010, 157–166.

2007) trains its SFCG translation model in a similar preprocessing stage.

By contrast, example-based machine translation (EBMT) approaches (Nagao, 1981, Sato and Nagao, 1990, Somers, 2003) are notable for their use of aligned parallel corpora at run time. EBMT research has successfully explored how various efficient data structures for pattern matching (Brown, 2004) can be leveraged to allow the decoder to access at decode-time portions of the training text that are most relevant to the text currently being translated. The Cunei machine translation toolkit (Phillips and Brown, 2009) is an open source, statistical EBMT system that follows this approach.

Suffix arrays are compact data structures which allow efficient pattern matching to be performed over all text in a corpus (Manber and Myers, 1990). Callison-Burch, Bannard, and Schroeder (2005) and Zhang and Vogel (2005) showed that suffix arrays can be adapted to allow phrase-based translation to calculate translation options for the translation model at run-time. A subsample of occurrences of given source phrase are used to calculate translation probabilities. By accessing the target corpus and word alignment data, the phrasal translations and their associated model parameters can be calculated at run-time. Lopez (2007) showed that hierarchical phrases can also by obtained at run time using a suffix array.

This article reports on an implementation of the basic techniques described in Lopez (2008) that was incoporated into the open source machine translation system Joshua (Li et al., 2009). The implementation described here enables users of Joshua to begin translating sentences using an aligned parallel corpus without having to extract an SCFG before decoding begins. The advantages of using this implementation are that any input sentence can be decoded (making it appropriate for live demos or real world use), and that the data structures require much less disk space than full phrase tables. This comes at the cost of slower running time for the decoder itself, since phrase translations have to be calculated on the fly.

2. Related Work

While example-based machine translation has long used corpus information at runtime, statistical phrase-based approaches typically include a preprocessing stage where an aligned parallel corpus is split into phrases, and parameter values are calculated for each phrase using simple relative frequency estimates. The phrase-based decoders Pharaoh (Koehn, 2004) and Moses (Koehn et al., 2007) take this approach, providing users with scripts to estimate a translation model from a sentence-aligned parallel corpus. Similarly, Hiero (Chiang, 2007) and the syntax-augmented machine translation (SAMT) system (Zollmann and Venugopal, 2006) both require a preprocessing stage to extract a SCFG translation model. Recent work in Moses (Huang and Koehn, p.c.) provides similar functionality for extracting an SCFG-based translation model during a preprocessing stage.

Callison-Burch, Bannard, and Schroeder (2005), Zhang and Vogel (2005), and Lopez (2008) all describe implementations of traditional phrase-based models extended to take advantage of suffix array data structures to extract phrase translation options at run-time. However, functional open source implementations of these have yet to be made available. Preliminary work has investigated integrating these techniques into Moses, but this work is not complete.[1]

Lopez (2008) provides a fast implementation of SCFG grammar extraction for Hiero which uses suffix arrays. This implementation allows Hiero to use an aligned parallel corpus at run-time in lieu of a pre-extracted SCFG. However, this implementation is not available as open source software due to intellectual property restrictions imposed by the University of Maryland. Cunei (Phillips and Brown, 2009)[2] is a statistical open source EBMT system that uses suffix arrays to extract relevant phrase pairs from an aligned parallel corpus at run-time.

3. Implementation: Data Structures and Algorithms

To extract hierarchical translation rules at run-time, the decoder must have access to the aligned parallel corpus. Internally, Joshua treats all source and target words as 32-bit integers. Each unique string that is encountered is assigned a unique integer. A hash map maintains the mapping from string to integer, while a corresponding list of strings maintains the mapping from integer back to string. Together these data structures comprise the **symbol table**.

A corpus can be considered a simple list whose size is equal to the number of words in the corpus. Using this approach with the symbol table, Joshua stores the source corpus as an integer array. An auxiliary array, with size equal to the number of sentences in the source corpus, is maintained. Elements in this auxiliary sentence array indicate the corpus index where each sentence begins. The target corpus is likewise represented by a **corpus array** and corresponding sentence array.

Once the source and target corpus arrays are available, the corresponding **suffix array**s can be constructed. Given a corpus array c and a symbol table, a second array is created of equal size to the corpus array. This array s is initialized such that $s[x] = x$. Where each integer in c represents a word string, each integer in s represents an index into c. The contents of s are sorted, using the element comparison function defined in Figure 1. After sorting, the indices of all instances in the corpus of any given phrase are located in a contiguous segment in the suffix array s.

While a phrase-based decoder can simply look up any required phrase in a suffix array, hierarchical decoders must deal with discontinuous phrases that include gaps. To deal with such phrases, Lopez (2008) defines an incremental algorithm for constructing a specialized trie (Fredkin, 1960) to represent the SCFG translation grammar. Given a source sentence, this algorithm constructs a **prefix tree with suffix links** by first

[1] Much of this preliminary work was conducted by Chris Callison-Burch, Andreas Eisele, Juri Ganitkevitch, and Adam Lopez at the Second Machine Translation Marathon in 2008.

[2] `http://sourceforge.net/projects/cunei`

```
function COMPARE_ELEMENTS($index1$, $index2$, $max$, $corpusEnd$)
    for $i = 0$; $i < max$; $i++$ do
        if $index1 + i < corpusEnd$ and $index2 + i > corpusEnd$ then
            return 1
        else if $index2 + i < corpusEnd$ and $index1 + i > corpusEnd$ then
            return $-1$
        else if $corpus[index1 + i]$ is lexicographically $<$ $corpus[index2 + i]$ then
            return $-1$
        else if $corpus[index1 + i]$ is lexicographically $>$ $corpus[index2 + i]$ then
            return 1
        end if
    end for
    return 0
end function
```

Figure 1. During suffix array creation, the contents of a corpus array are sorted using the element comparison function COMPARE_ELEMENTS

examining all possible contiguous source phrases, and uses the source suffix array to look up translations for contiguous phrases. Hierarchical phrases that consist of a contiguous phrase preceded or followed by a single nonterminal X can be constructed directly from the corresponding contiguous phrase. In this manner, the tree is gradually constructed into a grammar containing contiguous phrases and simple hierarchical phrases.

More complex hierarchical phrases are constructed using the QUERY_INTERSECT function in Figure 2. This function takes two smaller phrases $a\alpha$ and αb (a and b represent single words and α represents a sequence), along with the list of indices where these phrases are located. These two lists can be efficiently processed to determine the locations where the two phrases intersect to form the more complex phrase $a\alpha b$. In this way, all source hierarchical phrases can be located.

Each node in the prefix tree corresponds to a unique source phrase. Each node stores the complete list of all indices in the source corpus where that node's phrase occurs. These locations are used in conjunction with the target corpus array and the word alignment data to construct SCFG translation rules.

Ideally, once translation rules have been extracted for a given source phrase, those rules would be stored and not calculated again. Memory constraints typically dictate that not all rules are stored. Rather than storing the translation rules for a given source phrase at the corresponding node in the prefix tree, a single least-recently-used (LRU) cache is maintained. This cache maps from source phrase to the corresponding set of translation rules.

Another technique used to save memory is the option of using memory-mapped data

Algorithm Query_Intersect
Input: Sorted list of corpus locations matching source language pattern $a\alpha$: $M_{a\alpha}$
Input: Sorted list of corpus locations matching source language pattern αb: $M_{\alpha b}$

```
function Query_Intersect($M_{a\alpha}$, $M_{\alpha b}$)
    $M_{a\alpha b} \leftarrow \emptyset$                        ▷ Result list is initially empty
    $I \leftarrow |M_{a\alpha}|$         ▷ Number of instances of pattern $a\alpha$ in the source corpus
    $J \leftarrow |M_{\alpha b}|$         ▷ Number of instances of pattern $\alpha b$ in the source corpus
    $j \leftarrow 0$
    $i \leftarrow 0$
    while $i < I$ and $j < J$ do
                                                ▷ Ignore elements in $M_{\alpha b}$ that are
                                              ▷ too distant to intersect with $M_{a\alpha}[i]$
        while $j < J$ and $M_{a\alpha}[i] \ddot{>} M_{\alpha b}[j]$ do
            $j \leftarrow j + 1$
        end while
                                                     ▷ Verify that the corpus index
                                                     ▷ for the first terminal symbol
        $k \leftarrow i$                                       ▷ in pattern $\alpha b$ is the same
        while $M_{\alpha b}[i]_{,1} = M_{\alpha b}[k]_{,1}$ do          ▷ for locations $M_{\alpha b}[i]$ and $M_{\alpha b}[k]$
            $\ell \leftarrow j$
            while $\ell < J$ and not $M_{a\alpha}[i] \ddot{<} M_{\alpha b}[\ell]$ do
                if $M_{a\alpha}[i] \ddot{=} M_{\alpha b}[\ell]$ then
                    Intersect $M_{a\alpha}[i]$ with $M_{\alpha b}[\ell]$ and append result to $M_{a\alpha b}$
                end if
                $\ell \leftarrow \ell + 1$                     ▷ Proceed to next element in $M_{\alpha b}$
            end while
            $i \leftarrow i + 1$                       ▷ Proceed to next element in $M_{a\alpha}$
        end while
    end while
    return $M_{a\alpha b}$
end function
```

Result: Sorted list of corpus locations matching source language pattern $a\alpha b$: $M_{a\alpha b}$

Figure 2. Query intersection algorithm implemented in Joshua. This algorithm is adapted from a corrected version (Lopez, p.c.) of query intersection (Lopez, 2008).

structures. Memory-mapped version of the corpus array, suffix array, and alignment grids data structures are implemented and used by default.

4. Using Joshua

Given a word-aligned parallel corpus, the first step in extracting a grammar, either at run-time or during a preprocessing stage, is to compile the memory-mappable data structures to binary files on disk. The `joshua.corpus.suffix_array.Compile` program takes four parameters: source corpus text file, target corpus text file, word alignments text file, and an output directory path. The output directory, by convention, is named with the suffix `.josh`. This directory stores the binary representations of the symbol table, source and target corpus arrays, and the source and target suffix arrays. These binary files are given canonical names inside the `.josh` directory, so that the decoder can use them simply by specifying the `.josh` directory in the `tm_file=`... line of the Joshua configuration file.

It is often useful (especially during MERT) to extract a test set specific grammar once in a preprocessing step, since that test set will be translated many times and re-extracting the grammar each time would be wasteful. To perform this task, the program `joshua.prefix_tree.ExtractRules` can be used. When run directly, this program accepts either three arguments (a compiled `.josh` directory, file name for grammar to extract, and test file) or five arguments (source corpus text file, target corpus text file, word alignments text file, file name for grammar to extract, and test file). The following subsections document the mandatory and optional parameters that can be passed to this program through the `extractRules` ant task.

4.1. Mandatory parameters

testFile Path to plain text file containing a source language test file. The grammar extracted by `extractRules` will be all of the rules required to translate the sentences in this test file.

outputFile Path where extracted grammar will be placed. This grammar will consist of all of the rules required to translate the sentences in the test file defined in the `testFile` option.

joshDir Path to directory containing the binary files representing memory-mappable aligned parallel corpus.

The following parameters may be specified instead of the `joshDir` parameter.

sourceFileName Path to text file containing source corpus.
targetFileName Path to text file containing target corpus.
alignmentsFileName Path to text file containing word alignment data.

4.2. Optional parameters

maxPhraseSpan Defines the maximum span (in the source corpus) of any extracted SCFG rule. Default value is `10`.

maxPhraseLength Defines the maximum number of tokens (terminals plus nonterminals) allowed in the source right-hand side of any extracted SCFG rule. Default value is `10`.

maxNonterminals Defines the maximum number of nonterminal symbols allowed in the source side of any synchronous context-free rule extracted. Note: the number and type of nonterminals is the same in the source and target right-hand sides of a SCFG rule. Default value is `2`.

cacheSize Maximum number of source phrases for which translation rules will be maintained in the least-recently-used (LRU) cache.

encoding Defines the file encoding scheme of the input test file and the output grammar file. Default value is `UTF-8`.

ruleSampleSize When extracting SCFG rules for a given source language phrase, this option defines the number of instances of that source phrase will be sampled from the source training corpus for use in rule extraction. Default value is `300`.

startingSentence Defines the (1-based) sentence index in the test file where grammar extraction will begin. Default value is 1, indicating that a grammar will be extracted capable of translating sentences starting with the first sentence in the test file.

maxTestSentences Defines the number of sentences in the test file over which grammar extraction will be performed. Default value is `Integer.MAX_VALUE`. For example, given a test file of 100 sentences, the options `startingSentence="51"` `maxTestSentences="25"` would cause grammar extraction to be performed over test sentences 51–75.

The following parameters can be configured in the `extractRules` target to make rule extraction behave like the Hiero suffix array rule extractor (Lopez, 2008) instead of the behaving according to the rule extraction originally defined in Chiang (2005).

sentenceInitialX Boolean option indicates whether rules with an initial source-side nonterminal should be extracted from phrases at the start of a sentence, even though such rules do not have supporting corporal evidence. This option is provided for compatibility with Hiero's suffix array rule extractor (Lopez, 2008), in which this setting is set to true. Default value is `true`.

sentenceFinalX Boolean option indicates whether rules with a final source-side nonterminal should be extracted from phrases at the end of a sentence, even though such rules do not have supporting corporal evidence. This option is provided for compatibility with Hiero's suffix array rule extractor (Lopez, 2008), in which this setting is set to true. Default value is `true`.

edgeXViolates Boolean option indicates whether rules with an initial or final source-side nonterminal should be extracted when the source corpus phrase span for the rule, discounting the initial or final nonterminal, is already equal to the maximum phrase span value. Since nonterminals conceptually correspond to elided elements in the training corpus, setting this value to true allows phrases which have a longer phrase span than the maximum allowed phrase span. This option is provided for compatibility with Hiero's suffix array rule extractor (Lopez, 2008), in which this setting is set to true. Default value is `true`.

requireTightSpans Boolean option; if true, follow the heuristic from (Chiang, 2005): where multiple initial phrase pairs contain the same set of alignment points, consider only the smallest when performing rule extraction. For compatibility with Lopez (2008), set this parameter to false. Default value is `true`.

Additional options can be configured in the `extractRules` target to change the behavior of the prefix tree.

keepTree Boolean option indicates whether the prefix tree should persist from sentence to sentence during grammar extraction. If set to false, a new prefix tree will be created to process each sentence in the test file. Default value is `true`.

printPrefixTree Boolean option indicates whether a representation of the prefix tree should be printed to standard output. If set to true, the tree will be printed after processing each sentence in the test file. Default value is `false`.

Acknowledgements

This research was supported in part by the EuroMatrixPlus project funded by the European Commission under the Seventh Framework Programme, by the US National Science Foundation under grant IIS-0713448, and by the DARPA GALE program under contract number HR0011-06-2-0001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

Special thanks to Adam Lopez for his help and advice, and for making the LaTeXcode for his algorithms available.

Bibliography

Brown, Ralf D. 2004. A modified burrows-wheeler transform for highly-scalable example-based translation. In *Proceedings of the 6th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2004)*, Washington DC.

Callison-Burch, Chris, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, Ann Arbor, Michigan.

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

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

Fredkin, Edward. 1960. Trie memory. In *Communications of the ACM*, volume 3, pages 490–499.

Koehn, Philipp. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the 6th Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2004)*, Washington DC.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL-2007 Demo and Poster Sessions*, Prague, Czech Republic.

Koehn, Philipp, Franz Josef 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 (HLT/NAACL-2003)*, Edmonton, Alberta.

Kumar, Shankar, Yonggang Deng, and William Byrne. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75.

Li, Zhifei, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March. Association for Computational Linguistics.

Lopez, Adam. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Lopez, Adam. 2008. *Machine Translation by Pattern Matching.* Ph.D. thesis, University of Maryland, March.

Manber, Udi and Gene Myers. 1990. Suffix arrays: A new method for on-line string searches. In *The First Annual ACM-SIAM Symposium on Dicrete Algorithms*, pages 319–327.

Marcu, Daniel and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, Philadelphia, Pennsylvania.

Nagao, Makoto. 1981. A framework of a mechanical translation between Japanese and English by analogy principle. In A. Elithorn and R. Banerji, editors, *Artificial and Human Intelligence: edited review papers presented at the international NATO Symposium.* pages 173–180.

Och, Franz Josef and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Och, Franz Josef, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora.*

Phillips, Aaron B. and Ralf D. Brown. 2009. Cunei machine translation platform: System description. In *3rd Workshop on Example-Based Machine Translation*, Dublin, Ireland.

Sato, Satoshi and Makoto Nagao. 1990. Toward memory-based translation. In *Papers presented to the 13th International Conference on Computational Linguistics (CoLing-1990).*

Somers, Harold. 2003. An overview of EBMT. In Michael Carl and Andy Way, editors, *Recent Advances in Example-Based Machine Translation.* Kluwer Academic Publishers, chapter 4, pages 115–153.

Zhang, Ying and Stephan Vogel. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of the 10th Annual Conference of the European Association for Machine Translation (EAMT-2005)*, Budapest, Hungary.

Zollmann, Andreas and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the NAACL-2006 Workshop on Statistcal Machine Translation (WMT-06)*, New York, New York.